Quantum Information and Computation, Vol. 16, No. 9&10 (2016) 0862–0884

IMPROVED QUANTUM TERNARY ARITHMETIC

ALEX BOCHAROV[a]
Quantum Architectures and Computations Group, Microsoft Research
Redmond, Washington, 98052, USA

SHAWN X. CUI[b]
University of California
Santa Barbara, California, 93106, USA

MARTIN ROETTELER[c]
Quantum Architectures and Computations Group, Microsoft Research
Redmond, Washington, 98052, USA

KRYSTA M. SVORE[d]
Quantum Architectures and Computations Group, Microsoft Research
Redmond, Washington, 98052, USA

Received February 21, 2016
Revised May 9, 2016

Qutrit (or ternary) structures arise naturally in many quantum systems, notably in certain non-abelian anyon systems. We present efficient circuits for ternary reversible and quantum arithmetics. Our main result is the derivation of circuits for two families of ternary quantum adders. The main distinction from the binary adders is a richer ternary carry which leads potentially to higher resource counts in universal ternary bases. Our ternary ripple adder circuit has a circuit depth of $O(n)$ and uses only 1 ancilla, making it more efficient in both, circuit depth and width, when compared with previous constructions. Our ternary carry lookahead circuit has a circuit depth of only $O(\log n)$, while using $O(n)$ ancillas. Our approach works on two levels of abstraction: at the first level, descriptions of arithmetic circuits are given in terms of gates sequences that use various types of non-Clifford reflections. At the second level, we break down these reflections further by deriving them either from the two-qutrit Clifford gates and the non-Clifford gate $C(X) : |i, j\rangle \mapsto |i, j + \delta_{i,2} \bmod 3\rangle$ or from the two-qutrit Clifford gates and the non-Clifford gate $P_9 = \mathrm{diag}(e^{-2\pi i/9}, 1, e^{2\pi i/9})$. The two choices of elementary gate sets correspond to two possible mappings onto two different prospective quantum computing architectures which we call the metaplectic and the supermetaplectic basis, respectively. Finally, we develop a method to factor diagonal unitaries using multi-variate polynomials over the ternary finite field which allows to characterize classes of gates that can be implemented exactly over the supermetaplectic basis.

Keywords: Quantum circuits, ternary quantum systems, quantum adders

Communicated by: R Cleve & R Laflamme

[a] alexeib@microsoft.com
[b] cuixsh@gmail.com
[c] martinro@microsoft.com
[d] ksvore@microsoft.com

1 Introduction

Quantum computation has seen vast progress over the years, both theoretically and experimentally. Computations on a programmable and scalable fault-tolerant quantum computer will consist of fully controlled sequences of primitive operations such as unitary gates, measurements and state preparations. Such sequences are called *quantum circuits*. In the most commonly used circuit model, quantum information is stored in a collection of *qubits*, where each qubit has a two-dimensional Hilbert state space with the computational basis $\{|0\rangle, |1\rangle\}$. A standard universal gate set consists of Clifford gates and one non-Clifford gate such as the $\frac{\pi}{8}$-gate [1] or V-gate [2]. By design, circuits over a universal set can be used to approximate arbitrary quantum gates. Thus any quantum algorithm can be processed given a quantum computer with a universal gate set.

It has been noted by several researchers that architecture of certain quantum registers and gates is more naturally described by multi-valued logic as opposed to binary logic. History of experiments with ternary superconducting registers, in particular goes back to 1989 [3],[4]. More recently, in quantum computation domain, multi-valued logic has been proposed for linear ion traps [5], cold atoms [6], entangled photons [7]. It remains to be seen, at what scale it would be possible to balance out quantum universality and fault-tolerance in these and other similar architectures.

The research presented here is motivated in part by recent progress in circuit synthesis over universal quantum bases arising in topological quantum computing, where multi-qubit encoding is not necessarily the most natural choice. Several physical systems capable of performing topologically-protected quantum computations have a natural structure of a qutrit instead of a qubit, where a qutrit has a three-dimensional Hilbert space with the computational basis $\{|0\rangle, |1\rangle, |2\rangle\}$. For instance, in the $\mathrm{SU}(2)_4$ anyon system, anyons with quantum dimension $\sqrt{3}$ are well-suited for encoding quantum states in qutrits. What is more, it was shown in [8] that the $\mathrm{SU}(2)_4$ anyon system can be made universal through braiding and projective measurement of anyons. This anyonic structure is quite far from physical realization at the moment, yet, it offers a promise of comparatively simple quantum universality combined with native topological protection, which, in our opinion, makes it a worthwhile subject of forward-looking research.

In [9], an algorithm is given for approximation of any multi-qutrit gate with an asymptotically optimal circuit over the gate set Clifford + $\mathrm{diag}(1, 1, -1)$. This work also demonstrated the importance of *Householder reflections* for synthesis of efficient circuits. Even though the gate set turned out to be powerful enough for such synthesis, it had certain conceptual and practical limitations. Thus, it is quite unlikely that all the reversible classical permutation gates can be implemented exactly over Clifford + $\mathrm{diag}(1, 1, -1)$. This has a damping effect on implementation of arithmetic-heavy algorithms such as Shor's Factorization Algorithm, since the integer modular arithmetic is naturally described by reversible classical circuits. As a matter of principle such circuits may be represented exactly in commonly used multi-qubit circuit models. [e]

When compared to [9], the present paper aims at a more abstract level. Here we assume that the entire group of multi-qutrit classical permutations is representable at some cost, explore different scenarios of its representation and focus on synthesizing efficient circuits for ternary base arithmetic in these scenarios. Our thinking at this level remains reflection-centric. Previous research on non-binary reversible circuits [11] mostly focused on proving the universality of the local classical Clifford gates in combination with the *controlled-increment* gate $|j, k\rangle \mapsto |j, k + \delta_{j,d-1} \bmod d\rangle$, where d is the dimension of the qudit and δ is the Kronecker delta. Reversible circuits available in literature tend to

[e] To the extent the three-qubit Toffoli gate may be assumed exactly representable.

use ancillary qudits fairly liberally.

This paper differentiates itself from previous work in two ways. First, we explore several alternate methods for synthesizing classical reversible circuits. Second, we strive to minimize both the depth and the width of arithmetic circuits specifically. For example, we show in Section 3.1 that implementing of a faithful CARRY gate is not necessary in a correct ternary adder. By using a modified carry we eliminate the use of ancillary qutrits and reduce the cost of the gate when compared to a faithful CARRY as used in previous approaches to implement ternary carry ripple adders [12, 13, 14].

Our focus is mainly on two types of ternary quantum adders, a modified ripple-carry adder and a carry look-ahead adder. Both adders are generalized from their binary counterparts, but the generalizations are somewhat non-trivial. To add two n-qutrit numbers, the modified ripple-carry adder uses 1 ancilla and has a circuit depth of $O(n)$, while the carry look-ahead adder requires $O(n)$ ancillas and has a circuit depth of $O(\log n)$. Each of the two adders has an overall circuit size of $O(n)$ elementary gates. We also study various extensions of quantum adders including adder modulo 3^n, comparison, and subtraction.

We show these arithmetic circuits can be realized exactly using classical Clifford gates and one additional gate $C(X)$, the *controlled-increment* gate, whose matrix is given in Equation 1. $C(X)$ is a two-qutrit non-Clifford gate and it is universal for reversible classical computation. This sets the ternary reversible circuits apart from their binary analogs, where at least one three-qubit gate, e.g., the Toffoli gate, is required for universality.

$$C(X) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{1}$$

We also introduce a qutrit universal gate set Clifford + diag($e^{-\frac{2\pi i}{9}}, 1, e^{\frac{2\pi i}{9}}$), called the supermetaplectic basis, which resembles the single-qubit $\frac{\pi}{8}$-gate. Some techniques are developed to construct new quantum gates from old ones. As an application, it will be shown that all ternary arithmetic studied in this paper can be implemented exactly over the supermetaplectic basis.

We note that the reflection-centric synthesis of our adder circuits is a ternary counterpart of Toffoli-centric binary adder circuits as developed, for example, in [17] and [18]. This analogy is explained in more detail in corresponding sections throughout the paper. The exact representation of the $C(X)$ gate in supermetaplectic basis parallels the exact representation of the three-qubit Toffoli in the Clifford + $\frac{\pi}{8}$ basis. Quantitative comparison of the ternary and binary adders would be beyond the scope of this work. A major step towards comprehensive comparison of this kind was made in the upcoming paper [10] that demonstrates the advantages of emulating Shor's period funding function on ternary quantum computer and especially on the metaplectic topological quantum framework.

The paper is organized as follows. In Section 2, some preliminaries and notations used throughout the paper are given. In Section 3, we separately discuss the modified ripple-carry adder and carry look-ahead adder. Section 4 gives some extensions of quantum adders, including addition modulo 3^n,

comparison, and subtraction. Lastly in Section 5, we introduce the supermetaplectic basis and develop techniques for the construction of new gates.

2 Preliminaries and Notations

We denote the standard computational basis in a qutrit by $\{|0\rangle, |1\rangle, |2\rangle\}$. The terminology "qutrit" and "ternary" are sometimes used interchangeably. We call a quantum gate reversible or a classical *permutation gate* if it acts as some permutation of the standard basis elements. Unless otherwise noted, the arithmetic, e.g., addition, multiplication, etc., within a ket is assumed to be taken modulo 3. Also by default circuits are read from left to right, while compositions of gates when written as expressions follow the rule of matrix multiplications, i.e., they are read from right to left. Throughout the paper, the following ternary quantum gates are frequently used:

1. $X = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$, namely, $X|i\rangle = |i+1\rangle$.

2. $S_{0,1} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, namely, $S_{0,1}$ swaps $|0\rangle$ with $|1\rangle$ and fixes $|2\rangle$. Similarly, one can define $S_{0,2}, S_{1,2}$. This notation is also generalized to multi-qutrit gates. For instance, $S_{00,22}$ is a 2-qutrit gate, which swaps $|00\rangle$ with $|22\rangle$, and fixes all other basis elements.

3. Given an n-qutrit gate U, there are two versions of "controlled-U". The first version is called "soft-controlled-U," denoted by $\bigwedge(U)$, and is defined as the $(n+1)$-qutrit gate: $|i, j_1, \cdots j_n\rangle \mapsto (I \otimes U^i)|i, j_1, \cdots j_n\rangle$, where the first qutrit is called the control qutrit. The second version is the "hard-controlled-U" denoted by $C_c(U)$, where $c \in \{0, 1, 2\}$. The gate $C_c(U)$ is also an $(n+1)$-qutrit gate. However, in contrast to the soft-controlled-U, it maps $|i, j_1, \cdots j_n\rangle$ to $(I \otimes U^{\delta_{i,c}})|i, j_1, \cdots j_n\rangle$. It is direct to see that the $C_c(U)$'s for different c's are equivalent to each other up to some 1-qutrit reversible gates. Thus we also use $C(U)$ to denote a general $C_c(U)$. Moreover, the equality $\bigwedge(U) = C_1(U)(C_2(U))^2$ holds.

4. The following is a list of some important controlled gates:

   - $\mathsf{SUM} = \bigwedge(X) : |i, j\rangle \mapsto |i, i + j\rangle$,
   - $C(X) = C_c(X) : |i, j\rangle \mapsto |i, j + \delta_{i,c}\rangle$,
   - $\text{Horner} = \bigwedge(\bigwedge(X)) : |i, j, k\rangle \mapsto |i, j, ij + k\rangle$,
   - $C(\mathsf{SUM}) = C_c(\mathsf{SUM}) : |i, j, k\rangle \mapsto |i, j, j\delta_{i,c} + k\rangle$.

   The Horner gate is a qutrit generalization of the qubit Toffoli gate. See also [15] for additional background on the Horner gate.

5. $\mathsf{SWAP} : |i, j\rangle \mapsto |j, i\rangle$.

For graphical representations of the gates defined above, see Figure 1.

The qutrit Clifford group C [16] is generated by SUM, X, H, and Q, where H and Q are defined as follows:

$$H = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 1 & 1 \\ 1 & \zeta_3 & \zeta_3^2 \\ 1 & \zeta_3^2 & \zeta_3 \end{pmatrix}, \qquad Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \zeta_3 \end{pmatrix},$$

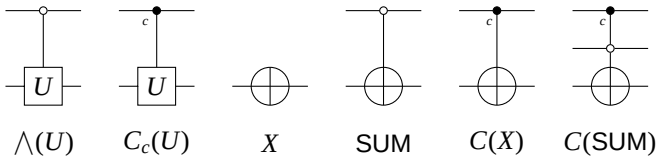

Fig. 1. Graphical representations of some ternary gates

where we use the notation $\zeta_n = e^{\frac{2\pi i}{n}}$ for $n \geq 1$.

It can be shown that, along with the SUM, all the reversible 1-qutrit gates and SWAP are also contained in C. Moreover, SUM and all the 1-qutrit reversible gates generate the subgroup of all reversible gates in C. Some other Clifford gates are Z and $\bigwedge(Z)$, where $Z = \mathrm{diag}(1, \zeta_3, \zeta_3^2)$, and $\bigwedge(Z) = (I \otimes H)\mathsf{SUM}(I \otimes H^{-1}) : |i, j\rangle \mapsto \zeta_3^{ij}|i, j\rangle$. However, $C(X)$, Horner, $C(\mathsf{SUM})$ and $S_{00,22}$ are non-Clifford gates.

Consider two pairs of standard basis vectors $|j_1\rangle$, $|k_1\rangle$ and $|j_2\rangle$, $|k_2\rangle$. It would be useful to note that the two-way classical reflection $S_{|j_1\rangle,|k_1\rangle}$ that swaps the $|j_1\rangle$, $|k_1\rangle$ and fixes everything else can be reduced to the corresponding reflection $S_{|j_2\rangle,|k_2\rangle}$ by applications of $O(n)$ SUM and SWAP gates (that are Clifford gates: see [9], Lemma 16). In particular, the two-way swap $S_{00,22}$ is Clifford-equivalent to any other two-qutrit two-way swap.

We think of Clifford gates as being *cheap* in the quantum sense. General rationale for this assumption is that such gates can be simulated classically. (Additional motivation coming from topological computing: in the context of non-abelian anyons such as $\mathrm{SU}(2)_4$ anyon system [8], Clifford gates can be obtained by anyon braiding alone.) Thus we define the complexity (resp. depth) of a circuit as the number (resp. depth) of non-Clifford gates.

The following two identities will be used, where $\omega(n)$ is the number of $1'$s in the binary expansion of n, and $\lfloor x \rfloor$ means the maximal integer less than or equal to x:

$$\sum_{i=1}^{\infty} \left\lfloor \frac{n}{2^i} \right\rfloor = n - \omega(n), \tag{2}$$

$$\sum_{i=1}^{\lfloor \log n \rfloor + 1} \left\lfloor \frac{n}{2^i} - \frac{1}{2} \right\rfloor = n - \lfloor \log n \rfloor - 1. \tag{3}$$

See also [17] for similar identities.

3 Quantum Ternary Adders

Given two n-trit numbers $a = a_{n-1} \cdots a_1 a_0$, $b = b_{n-1} \cdots b_1 b_0$, an adder computes their sum $s = s_n s_{n-1} \cdots s_0 = a + b$. The elementary method of adding two n-trit numbers is illustrated in Figure 1. Let $c_0 = 0$ be the initial carry trit and for $1 \leq i \leq n$, let c_i be the carry trit arising from $a_{i-1}, b_{i-1}, c_{i-1}$, namely, $c_i = 0$ if $a_{i-1} + b_{i-1} + c_{i-1} \leq 2$ and $c_i = 1$ otherwise. For $0 \leq i \leq n-1$, $s_i = a_i + b_i + c_i \bmod 3$ and $s_n = c_n$.

In Section 3.1 and Section 3.2, we present two methods to implement reversible ternary quantum adder: a ripple-carry adder and a carry look-ahead adder. The two adders are generalized from their binary counterparts [17, 18], but the generalizations are somewhat nontrivial, as seen later. On one hand, the modified ripple-carry adder uses only 1 ancilla for the whole process and has the circuit depth

	a_{n-1}	$\cdots$	a_1	a_0
	b_{n-1}	$\cdots$	b_1	b_0
c_n	c_{n-1}	$\cdots$	c_1	$c_0 = 0$
s_n	s_{n-1}	$\cdots$	s_1	s_0

Table 1. Addition of two n-trit numbers

in $O(n)$. On the other hand, the carry look-ahead adder requires $O(n)$ ancillas with the advantage of having circuit depth in $O(\log n)$. We will also compare the two adders to other ternary adders known in literature and show that our adders are more efficient both space-wise and depth-wise.

To implement the adders, we utilize $C(X)$, $C(\mathsf{SUM})$, $C(S_{0,1})$ and $S_{00,22}$ as the basic building units. As shown in Section 5.1, $C(\mathsf{SUM})$, $C(S_{0,1})$ and $S_{00,22}$ can all be constructed exactly from $C(X)$ and Clifford operations. Therefore, the circuit of adders can be designed from Clifford operations and $C(X)$ alone. The reason that we still treat $C(\mathsf{SUM})$, $C(S_{0,1})$ and $S_{00,22}$ as basic units is that it might be more efficient to synthesize them directly in some basis rather than breaking them up into $C(X)$'s. An example is the metaplectic basis [9], where $S_{00,22}$ has an efficient approximation by a metaplectic circuit.

3.1 Modified Ripple-Carry Adder

The binary quantum ripple-carry adder was considered in [19], where $O(n)$ ancillas are required to add two n-qubit numbers. In [17], the method was improved so that only 1 ancilla is necessary. Here we give a ternary generalization of the improved ripple-carry adder.

Note that in contrast to the binary case, the ternary carry is more complicated: if the inputs to a binary full adder are denoted by $a, b, c \in \mathbb{F}_2$, then the outgoing carry is given by $c_{out} = ab + ac + bc$, where all operations are computed modulo 2. In case of a ternary full adder with inputs $a, b, c \in \mathbb{F}_3$, the outgoing carry is given by $c_{out} = 2(1 + a + b + c)(ab + ac + bc) + abc$, where all operations are computed modulo 3. Though directly implementing this polynomial using the presented universal gates is possible, it leads to a relatively large number of elementary gates. A simple observation allows to reduce this cost significantly as it turns out that c_{out} does not have to be implemented for all 27 input triples but rather only 18 of them. Indeed, it can be shown inductively that—provided there is no initial incoming carry—for ternary adders, every carry trit c_i can only be either 0 or 1, but can never be 2. This is indicated also in Figure 2 where the crossed out case indicates that this can never occur in an actual addition: the case $c_{i+1} = 2$ is possible only if $c_i = 2$, which inductively we assume cannot happen. With this definition, c_{i+1} becomes a balanced function, i.e., there are the same number of inputs corresponding to each outcome c_{i+1}.

We sketch the idea of constructing the circuit to compute c_{i+1} from a_i, b_i and c_i based on this observation. As illustrated in Figure 2, c_{i+1} equals c_i for all but six inputs, the last three inputs in the column $c_{i+1} = 0$ and the last three in the column $c_{i+1} = 1$. For each of these six inputs, c_{i+1} equals $1 - c_i$. If the gate $S_{00,22}$ is applied to qutrits a_i, b_i, then the six inputs are turned into six new triples. See Figure 3 for the transition. Moreover, the new six triples are exactly equal to the set $\{(a, b, c) \in \{0, 1, 2\}^3 : a + b = c, c \neq 2\}$. In light of these observations, a reversible circuit, called Carry, is constructed, which takes c_i, a_i, b_i as input, and outputs c_{i+1} in the last qutrit. See Figure 2, where f and g are some functions of a_i, b_i, c_i. The exact shape of f and g is not important since they will be reversed at the appropriate step of the adder.

As illustrated in Figure 2, the circuit Carry is ancilla free, in contrast to the carry circuit considered

	$c_{i+1}=0$									$c_{i+1}=1$									~~$c_{i+1}=2$~~								
a_i	0	0	0	1	1	2	0	0	1	0	1	1	2	2	2	1	2	2	0	0	0	1	1	1	2	2	2
b_i	0	1	2	0	1	0	0	1	0	2	1	2	0	1	2	2	1	2	0	1	2	0	1	2	0	1	2
c_i	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	2	2	2	2	2	2	2	2	2

Table 2. Ternary carry table

	$c_{i+1}=0$			$c_{i+1}=1$		
a_i	0	0	1	1	2	2
b_i	0	1	0	2	1	2
c_i	1	1	1	0	0	0

$\xRightarrow{S_{00,22}}$

	$c_{i+1}=0$			$c_{i+1}=1$		
a_i	2	0	1	1	2	0
b_i	2	1	0	2	1	0
c_i	1	1	1	0	0	0

Table 3. Transition of inputs due to $S_{00,22}$

in [13] where 1 ancilla is required for each round of carry. See Figure 3 for the comparison. The circuit utilizes one $S_{00,22}$, one $C(S_{0,1})$, two SUM, and two SWAP gates. The SUM and SWAP are both Clifford gates, so only 2 non-Clifford gates are needed. The depth of Carry in terms of non-Clifford gates is also 2. Moreover, unlike the binary ripple-carry circuit MAJ [17] where the two qubits other than c_{i+1} end up with $a_i + b_i, c_i + b_i$, in our circuit the two qutrits other than c_{i+1} have the final values $f(a_i, b_i, c_i)$ and $g(a_i, b_i, c_i)$. This is the reason we call our carry circuit *modified*. However, as will be seen below, the modified carry circuit works in the same way as the regular one.

Let $C : |c_i, a_i, b_i\rangle \to |f(a_i, b_i, c_i), g(a_i, b_i, c_i), c_{i+1}\rangle$ be the Carry gate represented by the circuit in Figure 2. Similar to the adder circuit in [17], the modified ripple-carry adder circuit is designed in Figure 4, which, as an illustration, shows the addition of two 3-qutrit numbers.

In Figure 4, the qutrit c_0, initialized with 0, is the only ancilla required. The qutrit on the bottom holds the overflow trit, i.e., the highest trit in the sum. Therefore, to add two n-qutrit numbers, exactly 1 ancilla, n Carry gates, n inverse Carry gates and $2n$ SUM gates are required, and the depth of the circuit is $4n$. In contrast, the adder in [13] uses n ancillas and has the complexity in $O(n)$.

3.2 Carry Look-ahead Adder

In the ripple-carry adder, the carry c_{i+1} is computed only after the value of c_i has been obtained, and thus the overall depth of the circuit is in $O(n)$. One protocol to reduce the depth is the carry look-ahead adder studied in [18] for the binary addition. Here we generalize it to give a ternary carry look-ahead adder, which computes all the carry trits in depth $O(\log n)$ by introducing extra $O(n)$ ancillas.

The main idea is that there are relations between c_i and c_{i+1}, and more generally between c_i and c_j for $i \neq j$. For instance, if $a_i + b_i = 2$, then $c_{i+1} = c_i$. If $a_i + b_i = 1$, then $c_{i+1} = 0$ regardless of the value of c_i. See Figure 4 for a summary of the relation between c_{i+1} and c_i. Note that $c_0 = 0$, thus when $i = 0$, the column $c_{i+1} = c_i$ in Figure 4 becomes $c_1 = c_0 = 0$. Motivated by their relations, we define, for $0 \leq i < j \leq n$, the carry status indicator $C[i, j]$:

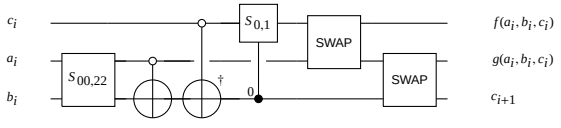

Fig. 2. the circuit Carry

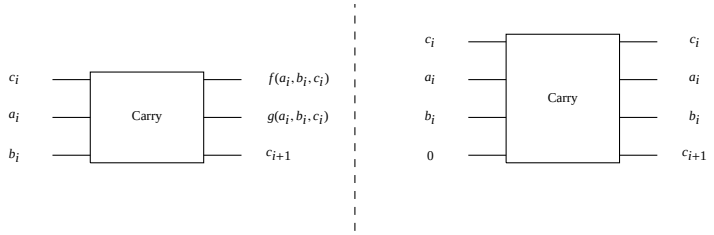

Fig. 3. (Left) ripple carry in the present paper; (Right) ripple carry studied in [13]

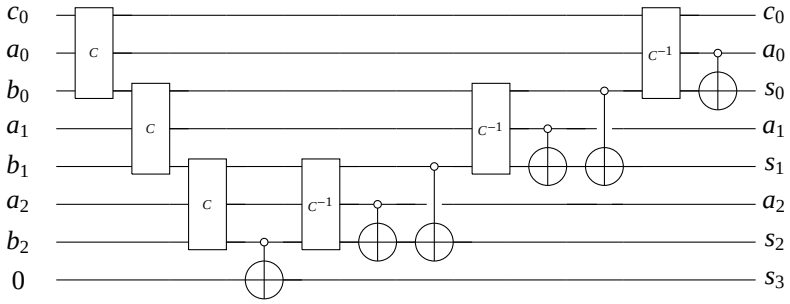

Fig. 4. Circuit for ripple-carry adder

	$c_{i+1} = 0$			$c_{i+1} = 1$			$c_{i+1} = c_i$		
a_i	0	0	1	1	2	2	0	1	2
b_i	0	1	0	2	1	2	2	1	0

Table 4. Relation between c_{i+1} and c_i

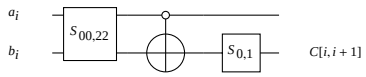

Fig. 5. Circuit AdjC computing $C[i, i+1]$, $0 < i < n$

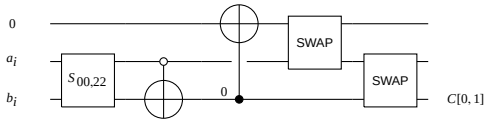

Fig. 6. Circuit AdjC_0 computing $C[0, 1]$

$$C[i, j] = \begin{cases} 0 & c_j = 0 \text{ regardless of } c_i \\ 1 & c_j = 1 \text{ regardless of } c_i \\ 2 & c_j = c_i \end{cases}$$

Since we already know $c_0 = 0$, the case $c_j = c_0$ is then the same as the first case $c_j = 0$. Thus we can treat these two cases as one, and design $C[0, j]$ so that it will never take the value 2, namely, we will have $C[0, j] = c_j$.

Explicitly, for $0 < i < n$, the circuit, AdjC, shown in Figure 5 computes $C[i, i+1]$ from a_i and b_i. It requires 1 non-Clifford gate $S_{00,22}$, and no ancilla. However, to compute $C[0, 1]$, we need to make use of 1 ancilla, and 2 non-Clifford gates $S_{00,22}, C(X)$. See Figure 6 for the circuit, which we call AdjC_0.

Having computed the carry status indicators for any two adjacent indices, we furthermore compute $C[i, j]$ for arbitrary $i \neq j$. For $0 \leq i < k < j \leq n$, $C[i, j]$ can be obtained from $C[i, k]$ and $C[k, j]$ by the *merging* formula in Figure 5.

Note that when $i = 0$, the row corresponding to $C[0, k] = 2$ in Figure 5 will never be used. Also when $C[0, k]$ takes values in $\{0, 1\}$, so will $C[0, j]$. A circuit, $\mathcal{M}$, realizing the *merging* formula is illustrated in Figure 7, where $\mathcal{M}$ takes $C[i, k], C[k, j]$, and an ancilla initialized to 0 as inputs, and outputs $C[i, j]$ to the ancilla. The circuit requires 1 non-Clifford gate $C(\mathsf{SUM})$.

The circuits AdjC and $AdjC_0$ both only depend on a_i and b_i, thus we can compute all the $C[i, i+1]'$s in one time slice. The nature of the *merging* formula enables us to obtain all the $C[0, j]'$s in $O(\log n)$ time slices. We elaborate this below.

For $i = 0, 1, \cdots, n-1$, let B_i be the working register configured to be $C[i, i+1]$ at the beginning, and let Z_{i+1} be the working registers initialized to $|0\rangle$, which will end up with $C[0, i+1]$. We also need $n - \omega(n) - \lfloor \log n \rfloor$ ancillas X_i initialized to $|0\rangle$. The circuit consists of three processes, namely, P-process, C-process, and P^{-1}-process. Each process roughly contains $\lfloor \log n \rfloor$ rounds.

In P-process, we compute all the carry status indicators of the form $C[2^t m, 2^t(m+1)]$ and write all the results into the ancillas, except the ones $C[0, 2^k]$ which are written to $Z[2^k]$. There are $\lfloor \log n \rfloor$

$\odot$		$C[k, j]$		
		0	1	2
	0	0	1	0
$C[i, k]$	1	0	1	1
	2	0	1	2

Table 5. The *merging* formula $C[i, j] = C[i, k] \odot C[k, j]$

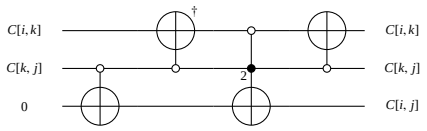

Fig. 7. Circuit $\mathcal{M}$ realizing the *merging* formula

$C[\lfloor \log \frac{n}{3} \rfloor]$	$\cdots$	$C[\lfloor \log n \rfloor - 3]$	$\cdots$	$C[0]$	
		$P^{-1}[\lfloor \log n \rfloor - 1]$	$\cdots$	$P^{-1}[2]$	$P^{-1}[1]$

Table 6. Parallelism between C- and P^{-1}-process

rounds, each $t = 1, \cdots, \lfloor \log n \rfloor$ corresponding to one round. In the t-th round, which we call the $P[t]$-round, the status indicators $C[2^t m, 2^t(m+1)]$, $m = 0, \cdots, \lfloor \frac{n}{2^t} \rfloor - 1$ are computed. By the *merging* formula, $C[2^t m, 2^t(m+1)]$ can be obtained from $C[2^{t-1}(2m), 2^{t-1}(2m+1)]$ and $[2^{t-1}(2m+1), 2^{t-1}(2m+2)]$, both of which have been computed in $P[t-1]$-round by induction. Moreover, the circuit $\mathcal{M}$ producing $C[2^t m, 2^t(m+1)]$ for different m's in the $P[t]$-round takes different carry status indicators in $P[t-1]$-round as input. Note that the $P[1]$-round requires the carry status indicators $C[i, i+1]'$s in the registers B_i. Therefore, in the $P[t]$-round, all the circuits $\mathcal{M}$ computing $C[2^t m, 2^t(m+1)]$ can be made parallel, and their inputs only depend on the carry status indicators from the $P[t-1]$-round. Thus, the depth of the circuit in P-process is $\lfloor \log n \rfloor$, the number of ancillas needed is $n - \omega(n) - \lfloor \log n \rfloor$, and the complexity is $n - \omega(n)$.

In C-process, we compute $C[0, j]$ into the register Z_j, $j = 1, \cdots, n$. This is performed in $\lfloor \log \frac{n}{3} \rfloor + 1$ rounds. Note that the $C[0, 2^k]'$s have already been obtained in P-process, and are located in the desired positions. For $t = \lfloor \log \frac{n}{3} \rfloor, \cdots, 0$, the $C[t]$-round consists of computing the carry status indicators $C[0, 2^t(2m+1)]$, $m = 1, \cdots, \lfloor \frac{n}{2^{t+1}} - \frac{1}{2} \rfloor$. Again, by the *merging* formula, we can get $C[0, 2^t(2m+1)]$ from $C[0, 2^{t+1}m]$ and $C[2^t(2m), 2^t(2m+1)]$. By induction, $C[0, 2^{t+1}m]$ has been obtained in earlier C-rounds if m is not a power of 2, and in the $P[t+1+\log m]$-round otherwise. Also $C[2^t(2m), 2^t(2m+1)]$ has been computed in the $P[t]$-round. Therefore, we can run all the $\mathcal{M}$ circuits in the $C[t]$-round in a parallel way. These circuits depend on the carry status indicators in the $P[t]$-round and $C[k]$-rounds, $k \geq t+1$. If m is a power of 2, then the corresponding $\mathcal{M}$ circuit also depends on $C[0, 2^{t+1}m]$ from the $P[t+1+\log m]$-round. Thus the circuit in C-process has a depth of $\lfloor \log \frac{n}{3} \rfloor + 1$, and the complexity is $n - \lfloor \log n \rfloor - 1$.

In P^{-1}-process, we set the ancillas back to $|0\rangle$, thus we need to reverse all the $\mathcal{M}$ circuits in P-process, except for those computing $C[0, 2^k]'$s which are not stored in the ancillas. The P^{-1}-process consists of $\lfloor \log n \rfloor - 1$ rounds. For $t = \lfloor \log n \rfloor - 1, \cdots, 1$, the $P^{-1}[t]$-round uncomputes $C[2^t m, 2^t(m+1)]$, $m = 1, \cdots, \lfloor \frac{n}{2^t} \rfloor - 1$ by using the inverse of $\mathcal{M}$. Note that in this process, all the $C[0, 2^k]'$s will not be touched. The process has a depth of $\lfloor \log n \rfloor - 1$, and the complexity of the circuit is $n - \omega(n) - \lfloor \log n \rfloor$.

We note that most parts of C-process and P^{-1}-process can actually be parallelized. The argument is as follows. All the inputs to the $C[t]$-round which are not of the form $C[0, 2^m]$ only depend on $C[k]$-rounds, $k \geq t+1$, and the $P[t]$-round. The inputs that are of the form $C[0, 2^m]$ were computed in $P[m]$-round, but they will not be touched in P^{-1}-process. The $P^{-1}[t+2]$-round only depends on the outputs in $P[t+1]$-round and $P[t+2]$-round. Thus the $C[t]$-round and the $P^{-1}[t+2]$-round can be performed simultaneously. The precise parallelism between C-process and P^{-1}-process is illustrated in Figure 6.

To summarize, the whole circuit uses $n - \omega(n) - \lfloor \log n \rfloor$ ancillas, and has a depth of $\lfloor \log n \rfloor + \left\lfloor \log \frac{n}{3} \right\rfloor + 2$. The total complexity of the circuit is $3n - 2\omega(n) - 2\lfloor \log n \rfloor - 1$.

3.3 Complete Circuit for Carry Look-Ahead Adder

We give two implementations of carry look-ahead adder, namely, the out-of-place adder and the in-place adder. Recall that the circuits in Figure 5, 6, and 7 are denoted by AdjC, AdjC_0, and $\mathcal{M}$, respectively. The complexity of both AdjC and $\mathcal{M}$ is 1, and the complexity of AdjC_0 is 2. The depth of these circuits is equal to their complexity.

3.3.1 Out-of-place Adder

Let A_i, B_i be the registers with initial value a_i, b_i, respectively, $i = 0, \cdots, n-1$. Let $Z_i, i = 0, \cdots, n$ be the registers initialized to be 0, which will hold the sum $a + b$ at the end of the computation. We need $n - \omega(n) - \lfloor \log n \rfloor$ ancillas X_i to store intermediate carry status indicators. The following is a description of the circuit of our out-of-place adder.

Out-of-place Procedure:

1. For $0 < i \leq n-1$, run the circuit AdjC on A_i, B_i, which outputs $C[i, i+1]$ to B_i. Run AdjC_0 on A_0, B_0, and Z_0 with Z_0 as the ancilla, which outputs $C[0,1]$ to B_0. Copy $C[0,1]$ to Z_1 with the SUM gate. The circuit has a depth of 2, and it consist of $n-1$ AdjC, 1 AdjC_0, and 1 SUM gates.

2. As discussed in Section 3.2, compute all the $C[0,i]$'s with the ancillas X_i's and the circuit $\mathcal{M}^{\pm 1}$. At the end of this process, the ancillas are returned to 0, and $Z_i = C[0,i], i = 1, \cdots, n$.[f] This requires $3n - 2\omega(n) - 2\lfloor \log n \rfloor - 1$ calls to the circuit $\mathcal{M}^{\pm 1}$, and has a circuit depth of $\lfloor \log n \rfloor + \left\lfloor \log \frac{n}{3} \right\rfloor + 2$.

3. Undo all the AdjC's and AdjC_0. At the end of this step, we have $B_i = b_i, Z_i = C[0,i] = c_i$. The circuit has a depth of 2, and it consist of $n-1$ AdjC^{-1}, 1 AdjC_0^{-1}, and 1 SUM^{-1}.

4. Set $Z_i = Z_i \oplus A_i \oplus B_i$, $0 \leq i \leq n-1$. This requires $2n$ SUM gates.

In summary, the out-of-place adder uses $n - \omega(n) - \lfloor \log n \rfloor$ ancillas, and has a circuit depth of $\lfloor \log n \rfloor + \left\lfloor \log \frac{n}{3} \right\rfloor + 6$, with the complexity of $5n - 2\omega(n) - 2\lfloor \log n \rfloor - 1$.

We represent AdjC_0, AdjC and $\mathcal{M}$ as shown in Figure 8. Their inverses are represented by the same circuit with $\triangleright$ replaced by $\triangleleft$. Also a black rectangle means the content will be changed after the application of the relevant gate, while a blank rectangle means the content remains the same. An an illustration, we give a complete out-of-place circuit for adding two 10-qutrit numbers in Figure 9, where we use x to stand for 10, and c_{ij} is the carry status indicator $C[i, j]$. From Figure 9, it is clear that the $C[0]$-round and $P^{-1}[2]$-round can be parallelized since the gates in these two rounds act on different wires. One can also verify the cost: the number of ancillas is $n - \omega(n) - \lfloor \log n \rfloor = 5$, the depth of the circuit is $\lfloor \log n \rfloor + \left\lfloor \log \frac{n}{3} \right\rfloor + 6 = 10$, and the complexity is $5n - 2\omega(n) - 2\lfloor \log n \rfloor - 1 = 39$.

[f] $Z_1 = C[0,1]$ was obtained in the previous step.

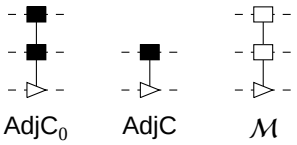

Fig. 8. Circuit glyphs for AdjC$_0$, AdjC and $\mathcal{M}$. The inverse gates AdjC$_0^{-1}$, AdjC^{-1} and $\mathcal{M}^{-1}$ are represented by mirror images of these glyphs.

3.3.2 In-place Adder

The idea of in-place adder is also generalized from that in [18]. Let $\overline{2}$ be the n-trit number with all 2's, namely $\overline{2} = 3^n - 1$. When no confusion arises, we make no distinction between a number and its trit representation. For two n-trit numbers a, b, denote by $a \oplus b$ the number obtained by trit-wise summation modulo 3, and denote by a' the number obtained by replacing every trit a_i by $2 - a_i$. Thus, the following equations hold:

$$a \oplus a' = \overline{2} \text{ and } a + a' = 3^n - 1.$$

Let $c = c_0 \cdots c_{n-1}$ be the sequence of the n low carry trits for a and b, and let s be the n low trits of $a + b$. Then we have

$$s = a + b \pmod{3^n} \text{ and } s = a \oplus b \oplus c.$$

Also note that $s' + a = 3^n - 1 - s + a = 3^n - 1 - b = b' \pmod{3^n}$.

Let $d = d_0 \cdots d_{n-1}$ be the n low carry trits resulting from adding s' and a. Then, $s' \oplus a \oplus d = b'$, and thus we have,

$$\begin{aligned} \overline{2} \oplus a \oplus b \oplus d &= s \oplus s' \oplus a \oplus b \oplus d \\ &= s \oplus b' \oplus b \\ &= \overline{2} \oplus a \oplus b \oplus c. \end{aligned}$$

Therefore, $c = d$, i.e., the n low carry trits for a, b are the same as those for s', a. We will use this property to implement the in-place adder.

For $0 \le i \le n - 1$, let A_i, B_i be the working registers initialized with a_i, b_i, respectively. We will need $2n - \omega(n) - \lfloor \log n \rfloor$ ancillas, n of which are denoted by $Z_0, Z_1, \cdots, Z_{n-1}$ and the rest are X_i's. Let Z_n be the working register which will store the high trit of $a + b$. All ancillas start with 0.

In-place Procedure:

1. As described in Out-of-place Procedure Step 1 through 3, compute all the carry trits $C[0, j]$ into Z_j, $j = 0, \cdots, n$. The ancillas X_i's and working registers A_i, B_i are all returned to their initial configuration at the end of the process. This has a circuit depth of $\lfloor \log n \rfloor + \left\lfloor \log \frac{n}{3} \right\rfloor + 6$, with the complexity of $5n - 2\omega(n) - 2\lfloor \log n \rfloor + 1$.

2. For $0 \le i \le n - 1$, let $B_i = B_i \oplus A_i \oplus Z_i$, namely, the register B_i's will store the n low trits of the sum $a + b$. This can be done by $2n$ SUM gates.

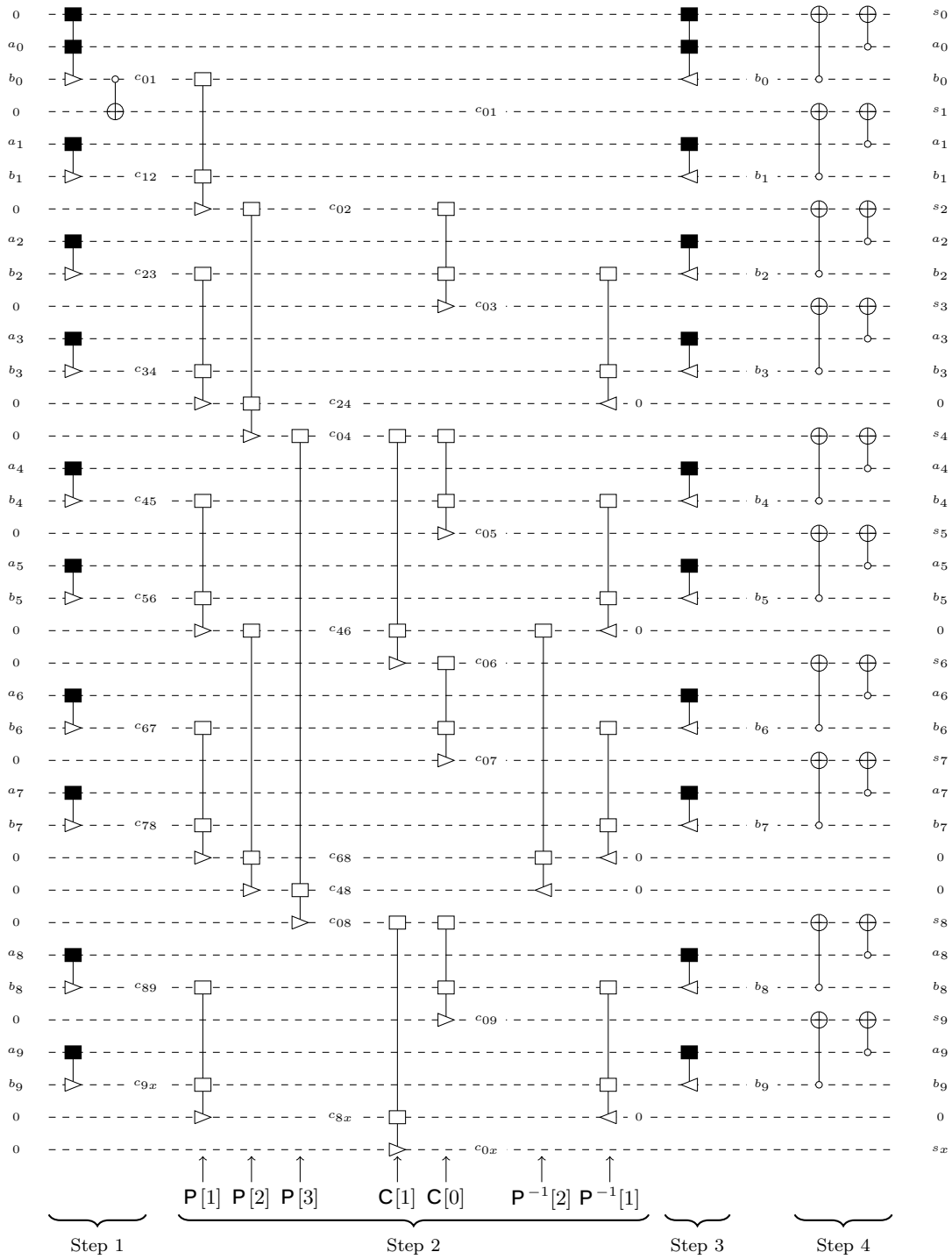

Fig. 9. Out-of-place carry look-ahead adder

3. Now we want to erase the n carry trits $C[0,i] = c_i$, $i = 0, \cdots, n-1$. For $0 \leq i \leq n-2$, let $B_i = 2 - B_i$. This can be achieved by $n-1$ $S_{0,2}$ gates.

4. Apply the inverse of the Out-of-place Procedure Step 1 through 3 on the registers A_i, B_i for $0 \leq i \leq n-2$ to erase the carry trits c_j stored in Z_j, $j = 0, \cdots, n-1$.

5. For $0 \leq i \leq n-2$, let $B_i = 2 - B_i$. Again this can be done by $n-1$ $S_{0,2}$ gates.

Tracing the cost of the circuit above, we see that the in-place adder has a depth of $\lfloor \log\, n \rfloor + \left\lfloor \log\, \frac{n}{3} \right\rfloor + \lfloor \log\,(n-1) \rfloor + \left\lfloor \log\, \frac{n-1}{3} \right\rfloor + 12$, and its complexity is $10n - 2\omega(n) - 2\lfloor \log\, n \rfloor - 2\omega(n-1) - 2\lfloor \log\,(n-1) \rfloor - 3$. Moreover, the number of ancillas required is $2n - \omega(n) - \lfloor \log\, n \rfloor$.

Figure 10 gives a complete circuit of in-place adder for $n = 10$. See Figure 8 and the last paragraph in Section 3.3.1 for the explanations of notations used in the circuit.

4 Extensions

In this section, we give various extensions based on the modified ripple-carry adder and the carry look-ahead adder, including addition modulo 3^n, subtraction, and comparison.

4.1 Addition Mod 3^n

To add two n-qutrit numbers modulo 3^n, we simply do not compute the the high carry trit c_n.

In the ripple-carry adder (see Figure 4), it suffices to remove the circuit C, SUM, C^{-1} in the middle, and the last qutrit on the bottom. Thus in total we need 1 ancilla, $2(n-1)$ Carry gates, and $2n-1$ SUM gates, and the depth of the circuit is $4(n-1)$.

In the out-of-place carry look-ahead adder, we run the circuit as described in Out-of-place Procedure in Section 3.3.1. However, in the first three steps of the procedure, we restrict the inputs to the $n-1$ low trits of a and b, namely, $a_0, \cdots, a_{n-2}, b_0, \cdots, b_{n-2}$, since there is no need to compute c_n. Of course, in the last step we still need to compute the modulo summation $a_i \oplus b_i \oplus c_i$ for all $0 \leq i \leq n-1$. Thus the out-of-place modulo adder uses $n - 1 - \omega(n-1) - \lfloor \log\,(n-1) \rfloor$ ancillas, and has a circuit depth of $\lfloor \log\,(n-1) \rfloor + \left\lfloor \log\, \frac{n-1}{3} \right\rfloor + 6$, with complexity $5(n-1) - 2\omega(n-1) - 2\lfloor \log\,(n-1) \rfloor + 1$.

Similarly, for the in-place carry look-ahead modulo 3^n adder, we run exactly the same circuit as the In-place Procedure in Section 3.3.2, except in Step 1 where we again restrict the inputs only to the $n-1$ low trits of a and b. It is direct to total the cost of the circuit. It has a depth of $2(\lfloor \log(n-1) \rfloor + \left\lfloor \log\, \frac{n-1}{3} \right\rfloor + 6)$, with the complexity of $2(5(n-1) - 2\omega(n-1) - 2\lfloor \log\,(n-1) \rfloor + 1)$. The number of ancillas required is $2(n-1) - \omega(n-1) - \lfloor \log\,(n-1) \rfloor$.

4.2 Subtraction

To compute $a - b$ for two n-trit numbers a, b, first convert a to a', then compute $a' + b$, and eventually convert $a' + b$ to $(a' + b)'$. Note that a' is the n-trit number obtained by replacing each a_i by $2 - a_i$, namely, $a' = 3^n - 1 - a$. Thus we have,

$$(a' + b)' = (3^n - 1 - a + b)' = 3^n - 1 - (3^n - 1 - a + b) = a - b.$$

Changing a to a' costs n Clifford gate $S_{0,2}$. Therefore, the circuit for subtraction has the same depth and complexity as the regular the adder.

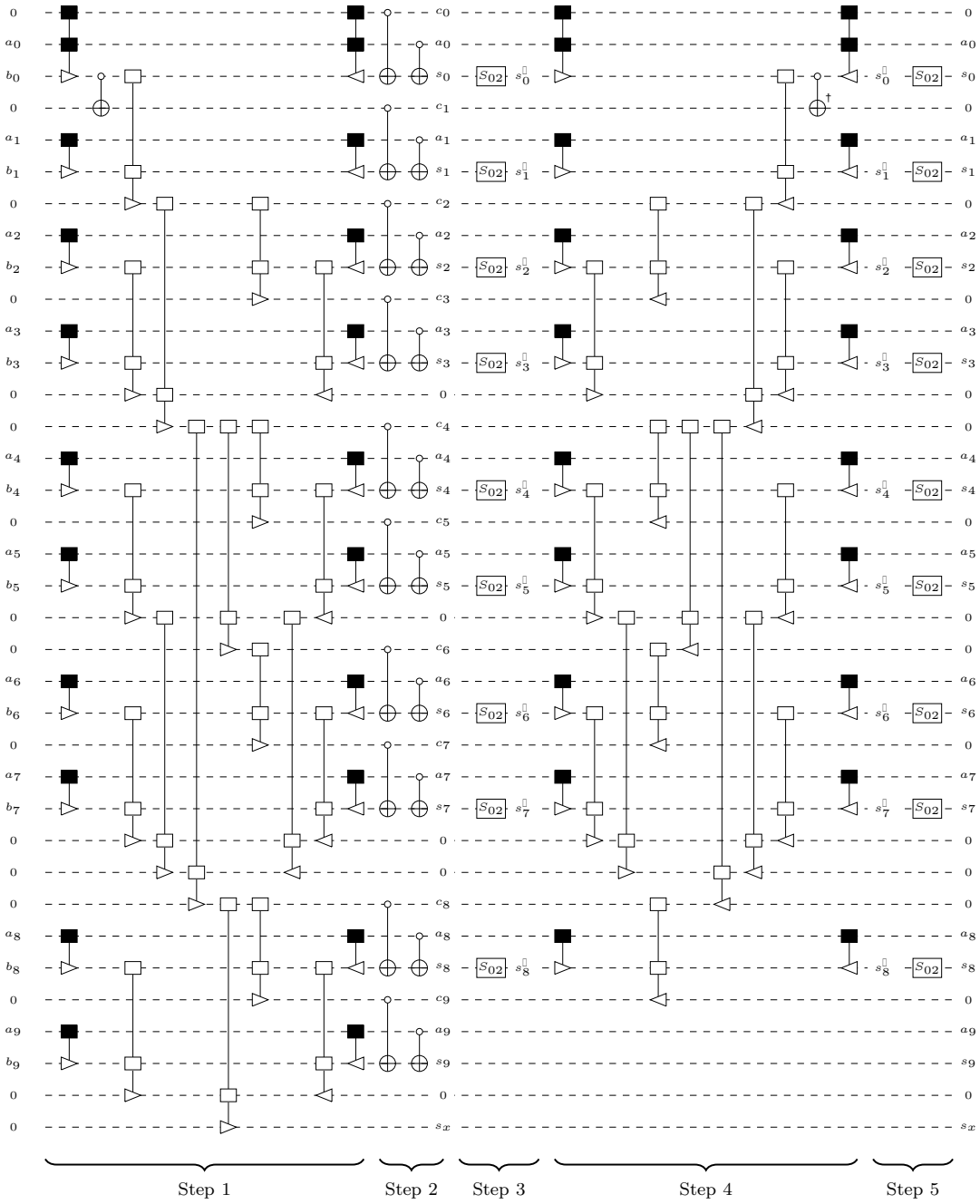

Fig. 10. In-place carry look-ahead adder

4.3 Comparison

Given the circuit for subtraction, it is straightforward to compare two numbers a and b. Actually, there is a circuit for the comparison of a, b with smaller complexity than that of subtraction since we only need to know the high trit of $a - b$. Let $a' = 3^n - 1 - a$, then $a - b \geq 0$ if and only if the high trit of $a' + b$ is 0.

In the ripple-carry adder, we convert a to a' and use the Carry gate C to compute all the carry trits $c_1, \cdots, c_n$ for $a' + b$. After copying c_n to the register storing the result of the comparison, we undo all the C's and convert a' back to a. The circuit thus requires 1 ancilla, $2n$ Carry gate C, 1 SUM gate, $2n$ $S_{0,2}$, and has a depth of $4n$.

In the carry look-ahead adder, again we first convert a to a'. To compute $a' + b$, the circuit sequentially generates all the carry status indicators $C[i, j]$'s. However, since we only care about the high trit $c_n = C[0, n]$, we can design a more efficient circuit to implement the comparison.

Recall from Section 3.2 that in P process we have obtained all the carry status indicators of the form $C[2^t m, 2^t(m + 1)]$, and in particular, any $C[0, 2^k]$ is of this form. Therefore, if $n = 2^k$ for some k, then c_n is obtained at the end of P process. At this moment, there is no need to go through the C process. Instead, we copy c_n into the register storing the result, and undo the P process. In general, let $k = \lceil \log n \rceil$, then we can just pad a and b by adding zeros in the front to make them 2^k-trit numbers, and use the circuit described above to compare a and b. We still call the 2^k-trit numbers a and b. For $0 \leq i \leq n - 1$, let $A_i = a_i, B_i = b_i$ be the working registers, and let R the register which will store the result of the comparison. We also need $2^k + 2(2^k - n)$ ancillas, among which $2(2^k - n)$ are used to hold the extra zeros in from of a and b, one is denoted by Z_0 as the ancilla to the AdjC_0 circuit, and the rest are denoted by X_i's.

Note that after padding a and b with zeros, the carry status indicators $C[i, j]$'s, $n \leq i < j \leq 2^k$, are known before the compilation, thus we can store their values in the registers and there is no need to recompute them later.

Carry Look-ahead Comparison:

1. Convert a to a'. This requires 2^k $S_{0,2}$ gates.
2. For $0 < i \leq n - 1$, run the circuit AdjC on A_i, B_i, which outputs $C[i, i + 1]$ to B_i. Run AdjC_0 on A_0, B_0, and Z_0 with Z_0 as the ancilla, which outputs $C[0, 1]$ to B_0. The circuit has a depth of 2, and it consist of $n - 1$ AdjC and 1 AdjC_0.
3. Perform the P process in Section 3.2 to compute all the $C[2^t m, 2^t(m + 1)]$ that are not known before compilation into the ancillary registers X_i. Note that here since we don't have the Z_i registers, all the $C[0, 2^m]$'s are also written to the X_i registers. The depth of the circuit is k, and the complexity is $2^k - \omega(2^k) - (2^k - n - \omega(2^k - n)) = n + \omega(2^k - n) - 1$.
4. Copy c_{2^k} to the result register R.
5. Undo Step 3.
6. Undo Step 2.
7. Undo Step 1.

Therefore, the total depth of the circuit above is $2k + 4 = 2\lceil \log n \rceil + 4$, and it has the complexity of $4n + 2\omega(2^k - n) = 4n + 2\omega(2^{\lceil \log n \rceil} - n)$. The number of ancillas used is $3 \cdot 2^{\lceil \log n \rceil} - 2n$.

5 Techniques for Constructing Quantum Gate Decompositions

In previous sections, we developed a system of ternary arithmetic with the focus on two types of quantum ternary adders. The building blocks of these circuits include the Carry circuit C, the circuits $\mathsf{AdjC}, \mathsf{AdjC}_0$ computing carry status indicators, and the *merging* formula $\mathcal{M}$. Moreover, the non-Clifford gates used in these four circuits are $S_{00,22}, C(S_{0,1}), C(X)$, and $C(\mathsf{SUM})$.

In this section, we show that it suffices to have $C(X)$ along with Clifford gates to produce the other three non-Clifford gates exactly. The key technique involved is to analyze the algebraic expressions of these gates. In Section 5.1, it is proven that $C(X)$ and Horner are equivalent up to Clifford gates, and that all other non-Clifford gates can be obtained from $C(X)$. In Section 5.2, we introduce a universal gate set called supermetaplectic basis, which is a qutrit analog of the qubit Clifford + $\frac{\pi}{8}$-gate. We then illustrate in Section 5.3 that $C(X)$ and Horner can both be implemented exactly over supermetaplectic basis. Therefore, with the supermetaplectic basis, the ternary circuits for arithmetic can be realized exactly.

5.1 Construction of Reversible Gates from Polynomial Expressions

Let $\mathbb{F}_3$ be the field with three elements $\{0, 1, 2\}$. Then any n-qutrit reversible gate can be represented as a map $\mathbb{F}_3^n \mapsto \mathbb{F}_3^n$, or a sequence of n functions $\mathbb{F}_3^n \mapsto \mathbb{F}_3$, if one identifies each $|i\rangle$ with i, $i = 0, 1, 2$. We will see that reversible gates have polynomial representations and these polynomial representations provide hints to construct one reversible gate from another.

Note that $0^2 = 0, 1^2 = 2^2 = 1 \pmod 3$, and thus $\delta_{i,0} = 1 - i^2 \pmod 3$. By default, arithmetic within a ket is taken modulo 3. The following is a list of polynomial expressions of some non-Clifford gates.

- $\mathsf{SUM} = \wedge(X) : |i, j\rangle \mapsto |i, i + j\rangle$;
- $C_0(X) : |i, j\rangle \mapsto |i, j + \delta_{i,0}\rangle = |i, j - i^2 + 1\rangle$;
- Horner:= $\wedge(\wedge(X)) : |i, j, k\rangle \mapsto |i, j, ij + k\rangle$;
- $C_0(\mathsf{SUM}) : |i, j, k\rangle \mapsto |i, j, k + (1 - i^2)j\rangle$.

The above list shows that if a qutrit works as a soft control, then it contributes a linear factor in the expression of the target qutrit, while a hard control qutrit contributes a quadratic factor.

Define $C'(X) : |i, j\rangle \mapsto |i, j + i^2\rangle$. Thus, $C'(X) = (I \otimes X)C_0(X)^{-1}$ is equivalent to $C(X)$. We will use $C'(X)$ below for the construction of other gates.

The relation between the expressions of Horner and $C'(X)$ resembles that of a bilinear form and a quadratic form, which are equivalent. This suggests that Horner and $C'(X)$ are also equivalent. Indeed, the following diagrams give a construction of one from another.

- implementation of Horner gate in terms of $C'(X)$: $|i, j, k\rangle \overset{\mathsf{SUM}_{1,2}}{\longrightarrow} |i, i + j, k\rangle \overset{C'(X)^{-1}_{2,3}}{\longrightarrow} |i, i + j, k - (i + j)^2\rangle \overset{\mathsf{SUM}^{-1}_{1,2}}{\longrightarrow} |i, j, k - i^2 - j^2 + ij\rangle \overset{C'(X)_{1,3}}{\longrightarrow} |i, j, k - j^2 + ij\rangle \overset{C'(X)_{2,3}}{\longrightarrow} |i, j, k + ij\rangle$.
- implementation of $C'(X)_{1,2}$ gate in terms of Horner : $|i, j, k\rangle \overset{\mathsf{SUM}_{1,3}}{\longrightarrow} |i, j, i + k\rangle \overset{\mathrm{Horner}_{1,3,2}}{\longrightarrow} |i, j + i^2 + ik, i + k\rangle \overset{\mathsf{SUM}^{-1}_{1,3}}{\longrightarrow} |i, j + i^2 + ik, k\rangle \overset{\mathrm{Horner}^{-1}_{1,3,2}}{\longrightarrow} |i, j + i^2, k\rangle$.

Note that in the construction of 2-qutrit $C'(X)$, we made use of a third qutrit, but that qutrit does not have to be clean, namely it could have arbitrary state.

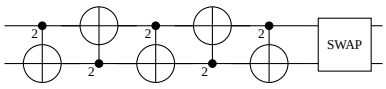

Fig. 11. A circuit for $S_{01,10}$

Similarly, $C'(X)$ is enough to construct $C(\mathsf{SUM})$:

$C_0(\mathsf{SUM})$: $|i,j,k\rangle \overset{C'(X)_{1,2}}{\longrightarrow} |i,i^2+j,k\rangle \overset{C'(X)_{2,3}}{\longrightarrow} |i,i^2+j,k+(i^2+j)^2\rangle \overset{C'(X)^{-1}_{1,2}}{\longrightarrow} |i,j,k+i^2+j^2-i^2j\rangle \overset{C'(X)^{-1}_{1,3}}{\longrightarrow} |i,j,k+j^2-i^2j\rangle \overset{C'(X)^{-1}_{2,3}}{\longrightarrow} |i,j,k-i^2j\rangle \overset{\mathsf{SUM}_{2,3}}{\longrightarrow} |i,j,k+(1-i^2)j\rangle$.

To implement $C(S_{0,1})$ and $S_{00,22}$, notice that the circuit in Figure 11 realizes $S_{01,10}$, and moreover we have:

- $S_{00,22} = \mathsf{SUM}^{-1}(X^{-1}\otimes I)S_{01,10}(X\otimes I)\mathsf{SUM}$.
- $C_0(S_{0,1}) = \mathsf{SUM}^{-1}_{2,1}(X^{-1}\otimes X^{-1})S_{00,22}(X\otimes X)\mathsf{SUM}_{2,1}$.

5.2 Supermetaplectic Basis

Recall from Section 2 that C is the qutrit Clifford group generated by H, Q, X, and SUM. Some other gates in C are Z and $\bigwedge(Z)$, where $Z = \mathrm{diag}(1,\zeta_3,\zeta_3^2)$, and $\bigwedge(Z) = (I\otimes H)\mathsf{SUM}(I\otimes H^{-1})$. It can be directly verified that $\bigwedge(Z)$ has the following expression:

$$\bigwedge(Z) : |i,j\rangle \mapsto \zeta_3^{ij}|i,j\rangle.$$

In [8], it has been established that the multi-qutrit *metaplectic* gate set $C + \mathrm{diag}(1,1,-1)$ or equivalently $C + \mathrm{diag}(1,\zeta_6,\zeta_6^2)$ was universal for quantum computation in the sense that any multi-qutrit unitary operator can be approximated to any given precision by a circuit over that gate set. We conjecture that the metaplectic gate set is not universal for *exact* reversible computation, i.e. it seems that the subgroup of reversible classical gates that can be represented exactly by metaplectic circuits is rather thin. In order to ensure exact representation of the reversible gates over a relatively simple multi-qutrit basis, we expand the basis by adding essentially the "cubic root" of the Z gate to it. To this end we increase the order of the root of unity used in defining the non-Clifford diagonal gate, and define P_9 as the 1-qutrit diagonal gate $\mathrm{diag}(\zeta_9^{-1},1,\zeta_9)$.[g]

Definition 1 *The gate set $C + P_9$ is called supermetaplectic basis.*

Since the P_9 gate is non-Clifford, this basis is universal for quantum computation. The supermetaplectic basis resembles the qubit Clifford + T basis in several aspects. Firstly, we show in Section 5.3 that all the reversible gates can be constructed exactly over the supermetaplectic basis. Secondly, the P_9 gate is a fundamental diagonal gate in the third level of the Clifford hierarchy [20]. Lastly, it was shown in [21] that P_9 can be obtained by magic state distillation.

5.3 Construction of Diagonal Gates from Polynomial Expressions

We continue exploring the use of polynomial expressions in constructing new quantum gates.

The group of reversible gates in C is generated by $\mathsf{SUM}, X, S_{1,2}$. More precisely, it is described by the following proposition.

[g] This is the the distillable gate denoted $M_3^\dagger$ in [21].

Proposition 2 *$\{S_{12}, X, \mathsf{SUM}\}$ generate a maximal subgroup, which is isomorphic to $\simeq \mathrm{GL}(n, \mathbb{F}_3) \rtimes \mathbb{F}_3^n$, of the group of reversible gates for any number n of qutrits.*

Proof: See Appendix 1. □

The statement in Proposition 2 for the case $n = 2$ was also proved in [9].

By the proof of Proposition 2, the correspondence between $\mathrm{GL}(n, \mathbb{F}_3) \rtimes \mathbb{F}_3^n$ and the group generated by $\{S_{12}, X, \mathsf{SUM}\}$ is as follows:

Given a pair $(A, v) \in \mathrm{GL}(n, \mathbb{F}_3) \rtimes \mathbb{F}_3^n$, where $A = (a_{ij})_{1 \le i,j \le n}$, $v = (v_i)_{1 \le i \le n}$, then the reversible n-qutrit gate corresponding to it maps $|x\rangle$, for any computational basis element $|x\rangle = |x_1, \cdots, x_n\rangle$, to $|A.x + v\rangle$. Moreover, any reversible gate of this form is generated by $\{S_{12}, X, \mathsf{SUM}\}$.

A function $f : \mathbb{F}_3^n \mapsto \mathbb{F}_3$ is called affine linear if $f(x_1, \cdots, x_n) = a_1 x_1 + \cdots + a_n x_n + b$, where $a_1, \cdots, a_n, b \in \mathbb{F}_3$. A reversible n-qutrit gate can be viewed as an n-tuple of functions: $|x\rangle \mapsto |f_1(x), \cdots, f_n(x)\rangle$, where we call f_i the coordinates of the gate. Then the above argument shows that a reversible n-qutrit gate is generated by $\{S_{12}, X, \mathsf{SUM}\}$ if and only if all of its coordinates are affine linear functions. Let $\mathcal{F}_n$ be the set of all affine linear functions from $\mathbb{F}_3^n$ to $\mathbb{F}_3$.

Let $\mathcal{D}$ be the group generated by the reversible gates in C, together with the diagonal gates $\bigwedge(Z)$ and P_9. We give a technique to characterize all the diagonal gates in $\mathcal{D}$.

By Proposition 2 and the argument above, the reversible gates in $\mathcal{D}$ can change the basis element $|x\rangle$ to any element of the form $|f_1(x), \cdots, f_n(x)\rangle$, where f_i is an affine linear function $\mathbb{F}_3^n$ to $\mathbb{F}_3$. The action of $\bigwedge(Z)$ and P_9 will contribute a scalar to the basis element. Thus the most general n-qutrit diagonal gate in $\mathcal{D}$ has the form:

$$|i_1, i_2, \cdots, i_n\rangle \mapsto \zeta_9^{\sum_{f \in \mathcal{F}_n} A_f f(i_1, \cdots, i_n)} \zeta_3^{\sum_{f,g \in \mathcal{F}_n} B_{f,g} f(i_1, \cdots, i_n) g(i_1, \cdots, i_n)} |i_1, i_2, \cdots, i_n\rangle, \tag{4}$$

where $A_f, B_{f,g}$ are integer parameters. Notice that the affine linear functions f and g take values in $\mathbb{F}_3$, while $A_f, B_{f,g}$ take values in $\mathbb{Z}$. We have to evaluate f, g first in $\{0, 1, 2\}$, then multiply by $A_f, B_{f,g}$ inside $\mathbb{Z}$. This is critical for the term ζ_9.

As an application, we show that $\bigwedge(\bigwedge(Z))$ and $C_2(Z)$ are both contained in $\mathcal{D}$. The expressions of relevant gates are given below.

- $\bigwedge(Z)|i, j\rangle = \zeta_3^{ij}|i, j\rangle, P_9|i\rangle = \zeta_9^{i}|i\rangle$,
- $X|i\rangle = |i + 1\rangle, S_{1,2}|i\rangle = |2i\rangle, \mathsf{SUM}|i, j\rangle = |i, i + j\rangle$.
- $\bigwedge(\bigwedge(Z)) : |i, j, k\rangle \mapsto \zeta_3^{ijk}|i, j, k\rangle$.
- $C_2(Z) : |i, j\rangle \mapsto \zeta_3^{j\delta_{i,2}}|i, j\rangle$.

For $n = 3$, the coefficient in Formula 4 can be written as:

$$L(i, j, k) = \zeta_9^{\sum_{a,b,c,d=0}^{2} A_{a,b,c,d}(ai+bj+ck+d)} \zeta_3^{Bij+Cjk+Dik}, \quad i, j, k \in \mathbb{F}_3, \tag{5}$$

where $A_{a,b,c,d}, B, C, D$ are integer parameters [h]. Again $ai + bj + ck + d$ is assumed to be taken modulo 3.

To construct $\bigwedge(\bigwedge(Z))$, set $L(i, j, k) = \zeta_3^{ijk}$. Since $\zeta_9 = \zeta_3^3$, we get the equation:

[h] Actually there are also terms i^2, j^2, k^2 on the exponent of ζ_3, but it is direct to see that $\zeta_3^{i^2} = \zeta_9^{(2i \bmod 3) - ((2-i) \bmod 3)}$ up to a global phase, so the square terms can be absorbed into the ζ_9 terms.

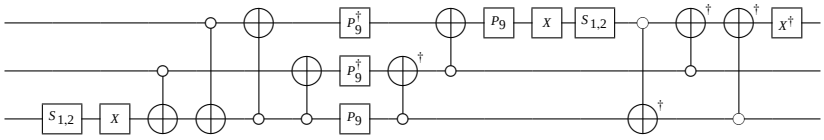

Fig. 12. A circuit for $\bigwedge(\bigwedge(Z))$

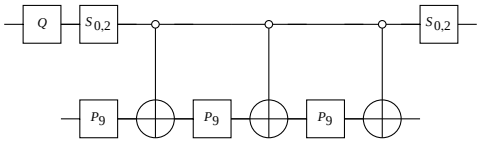

Fig. 13. A circuit for $C_2(Z)$

$$\text{Equ}(i,j,k): \sum_{a,b,c,d} A_{a,b,c,d}(ai+bj+ck+d)+3(Bij+Cjk+Dik) = 3ijk \,(\text{ mod } 9), \quad i,j,k \in \mathbb{F}_3. \quad (6)$$

The set $\{\text{Equ}(i,j,k) : i,j,k \in \mathbb{F}_3\}$ is a system of 27 linear equations in the variables $A_{a,b,c,d}, B, C$, and D. Thus there is an efficient way to find the solutions, if any.

By direct calculations, one solution to the above system of equations is:

$$\zeta_3^{ijk} = \zeta_9^{(1+2i+j+k)+2(1+2i+j+2k)+6(2+2i+j+2k)+2(1+2i+2j+k)+6(2+2i+2j+k)+4(1+2i+2j+2k)+6(2+2i+2j+2k)}, \quad (7)$$

where the terms on the exponent within each parenthesis is taken modulo 3.

In light of the solution in Equation 7, it is not hard to create a circuit realizing $\bigwedge(\bigwedge(Z))$. Explicitly, this is given in Figure 12.

Similarly, with the same method, we construct a circuit for $C_2(Z)$. See Figure 13.

Note that $\bigwedge(\bigwedge(Z)), C_2(Z)$ are related with Horner, $C_2(X)$, respectively, by the Clifford gate H, namely, we have,

- $(I \otimes H)C_2(X)(I \otimes H^\dagger) = C_2(Z)$
- $(I \otimes I \otimes H)\text{Horner}(I \otimes I \otimes H^\dagger) = \bigwedge(\bigwedge(Z))$.

Therefore, both Horner and $C_2(X)$ can be implemented exactly over supermetaplectic basis.

Remark 3 *1. The papers [22, 23] developed a similar framework for the binary case.*

2. If one uses the similar technique for the qubit Clifford + T gates, namely replacing (ζ_9, ζ_3) *with* $(\zeta_8, -1)$*, one obtains a circuit for the Toffoli gate with T-depth* 3*, which is optimal in the ancilla free scenario.*

6 Conclusion

We developed improved ternary circuits for reversible ternary adders of two types: the modified ripple-carry and the carry look-ahead adder. We have also derived solutions for a modulo 3^n adder, subtraction and comparison in ternary encoding. We have offered two levels of abstraction for describing the

corresponding ternary circuits: one in terms of reversible reflections of certain types and one in a more uniform language that allows only one non-Clifford gate: either the $C(X) : |i, j\rangle \mapsto |i, j + \delta_{i,2} \bmod 3\rangle$ or the $P_9 = \mathrm{diag}(e^{-2\pi i/9}, 1, e^{2\pi i/9})$ gate.

Future circuit synthesis work should entail the design of fully modular adders, circuits for singly- and doubly-controlled adders, as well as optimized circuits for singly- and doubly-controlled additive shifts that would be essential parts of Shor's integer factorization algorithm.

An important theoretical direction of future work would be establishing lower complexity bound for the arithmetic circuits and evaluating the efficiency of designs presented here versus these bounds.

Acknowledgment

Most of the work in the present paper was done during Summer 2015 when the second author was interning with Microsoft QuArC Group.

References

1. Boykin, P Oscar and Mor, Tal and Pulver, Matthew and Roychowdhury, Vwani and Vatan, Farrokh: A new universal and fault-tolerant quantum basis. Information Processing Letters,75(3):101–107, 2000
2. Harrow, Aram W and Recht, Benjamin and Chuang, Isaac L: Efficient discrete approximations of quantum gates. Journal of Mathematical Physics, 43(9), 4445–4451, 2002
3. Morisue, Mititada and Oochi, Kiyoshi and Nishizawa, Mitsuhiro: A novel ternary logic circuit using Josephson junction. IEEE Trans. Magn., 25(2), 1989
4. Morisue, Mititada and Endo, Jun and Morooka, Toshimitu and Shimizu, Nobuhiro and Sakamoto, Masahiro: A Josephson ternary memory circuit. Multiple-Valued Logic, 1998. Proceedings. 1998 28th IEEE International Symposium on, 19 – 24, 1998
5. Muthukrishnan, Ashok and Stroud, Carlos R, Jr.: Multivalued logic gates for quantum computation. Phys. Rev. A., 62(5), 051309, 2000
6. Smith, Aaron and Anderson, Brian E and Sosa-Martinez, Hector and Riofrio, Carlos A and Deutsch, Ivan H and Jessen, Poul S: Quantum control in the Cs $6S_{1/2}$ ground manifold using rf and μw magnetic fields. Phys. Rev. Lett., 111(170502), 2013
7. Malik, Mehul and Erhard, Manuel and Huber, Marcus and Krenn, Mario and Fickler, Robert, Zeilinger, Anton: Multi-photon entanglement in high dimensions, Nature Photonics 10, 248–252, 2016
8. Cui, Shawn X and Wang, Zhenghan: Universal quantum computation with metaplectic anyons. Journal of Mathematical Physics, 56(3), 032202, 2015
9. Bocharov, Alex and Cui, Xingshan and Kliuchnikov, Vadym and Wang, Zhenghan: Efficient topological compilation for weakly-integral anyon model. Phys. Rev. A 93, 012313, 2016
10. Bocharov, Alex and Roetteler, Martin and Svore, Krysta M.: Factoring with Qutrits: Shor's Algorithm on Ternary and Metaplectic Quantum Architectures. (In preparation)
11. Brennen, Gavin K and Bullock, Stephen S and O'Leary, Dianne P: Efficient circuits for exact-universal computations with qudits. Quantum Information and Computation, 6, 436, 2006
12. Miller, D Michael and Dueck, Gerhard W and Maslov, Dmitri: A synthesis method for MVL reversible logic. 34th IEEE International Symposium on Multiple-Valued Logic (ISMVL), 74–80, 2004
13. Satoh, Takahiko and Nagayama, Shota and Van Meter, Rodney: A reversible ternary adder for quantum computation. Asian Conf. on Quantum Information Science, 2007
14. Khan, Mozammel HA and Perkowski, Marek A: Quantum ternary parallel adder/subtractor with partially-look-ahead carry. Journal of Systems Architecture, 53(7), 453–464, 2007
15. Grassl, Markus and Roetteler, Martin and Beth, Thomas: Efficient quantum circuits for non-qubit quantum error-correcting codes. International Journal of Foundations of Computer Science, 14(5), 757–775, 2003
16. Gottesman, Daniel: Fault-tolerant quantum computation with higher-dimensional systems. Quantum Computing and Quantum Communications, Springer, 302–313, 1999
17. Cuccaro, Steven A and Draper, Thomas G and Kutin, Samuel A and Moulton, David Petrie: A new quantum ripple-carry addition circuit. arXiv:quant-ph/0410184, 2004

18. Draper, Thomas G and Kutin, Samuel A and Rains, Eric M and Svore, Krysta M: A logarithmic-depth quantum carry-lookahead adder. Quantum Information and Computation, 6(4), 351–369, 2006
19. Vedral, Vlatko and Barenco, Adriano and Ekert, Artur: Quantum networks for elementary arithmetic operations. Phys. Rev. A, 54(1), 147, 1996
20. Howard, Mark and Vala, Jiri: Qudit versions of the qubit $\pi/8$ gate. Phys. Rev. A, 86(2), 022316, 2012
21. Campbell, Earl T and Anwar, Hussain and Browne, Dan E: Magic-state distillation in all prime dimensions using quantum reed-muller codes. Phys. Rev. X, 2(4), 041021, 2012
22. Amy, Matthew and Maslov, Dmitri and Mosca, Michele and Roetteler, Martin: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 32(6), 818–830, 2013
23. Amy, Matthew and Maslov, Dmitri and Mosca, Michele: Polynomial-time T-depth optimization of Clifford+T circuits via matroid partitioning. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 33(10), 1476–1489, 2014
24. Scott, Leonard L: Representations in characteristic p. The Santa Cruz Conference on Finite Groups (Univ. California, Santa Cruz, Calif., 1979, 37, 319–331, 1980
25. Liebeck, Martin W and Praeger, Cheryl E and Saxl, Jan: On the O'Nan-Scott theorem for finite primitive permutation groups. Journal of the Australian Mathematical Society (Series A), 44(03), 389–396, 1988

Appendix A Reversible gates generated by $\{S_{12}, X, \mathsf{SUM}\}$

Proposition A.1 *$\{S_{12}, X, \mathsf{SUM}\}$ generate a maximal subgroup, which is isomorphic to $\simeq \mathrm{GL}(n, \mathbb{F}_3) \rtimes \mathbb{F}_3^n$, of the group of reversible gates (the permutation group) for any number n of qutrits.*

Proof: Let $\mathbb{F}_3^n$ be the n-dimensional vector space over the finite field $\mathbb{F}_3$. Then there is a one-to-one correspondence between the elements of $\mathbb{F}_3^n$ and the computational basis of the n-qutrit space $(\mathbb{C}^3)^{\otimes n}$. That is, any element $(x_1, \cdots, x_n) \in \mathbb{F}_3^n$ corresponds to the basis element $|x_1, \cdots, x_n\rangle$. Thus any automorphism on $\mathbb{F}_3^n$ induces a permutation on the n-qutrit basis, which is a reversible n-qutrit gate.

Let $G = \mathrm{GL}(n, \mathbb{F}_3) \rtimes \mathbb{F}_3^n$, the semidirect product of $\mathrm{GL}(n, \mathbb{F}_3)$ and $\mathbb{F}_3^n$, and let S_{3^n} be the symmetric group on 3^n elements, or equivalently the group of reversible gates on n qutrits. We first prove the group generated by $\{S_{12}, X, \mathsf{SUM}\}$ is isomorphic to G. As a corollary of applying the O'Nan-Scott Theorem to the classification of maximal subgroups of the symmetric group [24] [25], it follows that G is a maximal subgroup of S_{3^n}.

The group G is the affine linear group of degree n over $\mathbb{F}_3$, namely, it consists of all the pairs (A, v), where A is an $n \times n$ invertible group with entries in $\mathbb{F}_3$, and v is a vector in $\mathbb{F}_3^n$. The group G acts on $\mathbb{F}_3^n$ as follows:

$$(A, v).x = A.x + v, \quad (A, v) \in G, x \in \mathbb{F}_3^n$$

Therefore, we get a map $\varphi : G \longrightarrow U(3^n)$, such that $\varphi(A, v)|x\rangle = |Ax + v\rangle$, where $|x\rangle$ is any computational basis vector. This map φ is apparently a group homomorphism and injective.

For $1 \leq i \neq j \leq n$, define $A_{ij}, M_i \in \mathrm{GL}(n, \mathbb{F}_3), v_i \in \mathbb{F}_3^n$ as follows.

$$A_{ij} = I_n + E_{ji} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & \ddots & & & \\ & & 1 & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}, \quad M_i = I_n + E_{ii} = \mathrm{diag}(1, \cdots, 1, 2, 1, \cdots, 1), \quad v_i = (0, \cdots, 0, 1, 0, \cdots, 0).$$

It is straightforward to check that $\varphi(A_{ij}, 0) = \mathsf{SUM}_{ij}$, $\varphi(M_i, 0) = (S_{1,2})_i$, $\varphi(0, v_i) = X_i$, where the subscript of the gate on the right hand side of each expression denotes the qutrits it acts on. For instance, X_i is the X gate acting on the i-th qutrit. Therefore, the group generated by $\mathsf{SUM}, X, S_{1,2}$ is isomorphic to the group generated by A_{ij}, M_i, v_i, for $1 \leq i \neq j \leq n$.

Clearly all the v_i's generate $\mathbb{F}_n^3$ as an additive group. We next prove that A_{ij}, M_i generate the group $\mathrm{GL}(n, \mathbb{F}_3)$.

Let $B_{ij} = M_i A_{ij} A_{ji}^{-1} A_{ij} = I_n - E_{ii} - E_{jj} + E_{ij} + E_{ji}$, thus B_{ij} swaps the two basis elements e_i and e_j. Now given any matrix $A \in \mathrm{GL}(n, \mathbb{F}_3)$, multiplying A on the left by A_{ij}, B_{ij}, and M_i constitutes the three types of row operations on A, and since A is invertible, it can always be reduced to the identity matrix by row operations. This proves that any matrix in $\mathrm{GL}(n, \mathbb{F}_3)$ can be written as a product of A_{ij}, B_{ij}, and M_i. Therefore, $\mathrm{GL}(n, \mathbb{F}_3)$ is generated by A_{ij}, M_i, and hence G is generated by A_{ij}, M_i, and v_i.

Combining the above argument, we showed that the group generated by SUM, S_{12}, X is isomorphic to $G = \mathrm{GL}(n, \mathbb{F}_3) \rtimes \mathbb{F}_3^n$. □