Quantum Information and Computation, Vol. 13, No. 9&10 (2013) 0771-0792

REVERSIBLE LOGIC SYNTHESIS BY QUANTUM ROTATION GATES

AFSHIN ABDOLLAHI* MEHDI SAEEDI† MASSOUD PEDRAM
Department of Electrical Engineering, University of Southern California
Los Angeles, CA 90089-2562

Received November 12, 2012
Revised February 17, 2013

A rotation-based synthesis framework for reversible logic is proposed. We develop a canonical representation based on binary decision diagrams and introduce operators to manipulate the developed representation model. Furthermore, a recursive functional bi-decomposition approach is proposed to automatically synthesize a given function. While Boolean reversible logic is particularly addressed, our framework constructs intermediate quantum states that may be in superposition, hence we combine techniques from reversible Boolean logic and quantum computation. The proposed approach results in quadratic gate count for multiple-control Toffoli gates without ancillae, linear depth for quantum carry-ripple adder, and quasilinear size for quantum multiplexer.

Keywords: Quantum circuits, reversible circuits, rotation gates, binary decision diagrams
Communicated by: R Cleve & A Harrow

1 Introduction

The appeal for research on quantum information processing [1] is due to three major reasons. **(1)** Working with information encoded at the atomic scale such as ions and even elementary particles such as photons is a scientific advance. **(2)** Direct manipulation of quantum information may create new capabilities such as ultra-precise measurement [2], telemetry, and quantum lithography [3], and computational simulation of quantum-mechanical phenomena [4]. **(3)** Some time-exponential computational tasks with non-quantum input and output have efficient quantum algorithms [1]. Particularly, most quantum circuits achieve a quantum speed-up over conventional algorithms [5]. However, useful applications remain limited.

Recent advances in fault-tolerant quantum computing decrease per-gate error rates below the threshold estimate [6] promising larger quantum computing systems. To be able to do efficient quantum computation, one needs to have an efficient set of computer-aided design tools in addition to the ability of working with favorable complexity class and controlling quantum mechanical systems with a high fidelity and long coherence times. This is comparable with the classical domain where a Turing machine, a high clock speed and no errors in switching were not adequate to design fast modern computers.

Quantum circuit design with algorithmic techniques and CAD tools has been followed by several researchers. The proposed methods either addressed permutation matrices [7]

*Current address: Knight Capital Americas LLC, 545 Washington Blvd, Jersey City, NJ 07310
†Corresponding author: msaeedi@usc.edu.

or unitary matrices, e.g., [8]. Permutation matrices and reversible circuits are an important class of computations that should be efficiently performed for the purpose of efficient quantum computation. Indeed, Boolean reversible circuits have attracted attention as components in several quantum algorithms including Shor's quantum factoring [9, 10] and stabilizer circuits [11].

In this paper, a canonical decision diagram-based representation is presented with novel techniques for synthesis of circuits with binary inputs. This work may be considered along with the work done for the synthesis of reversible circuits [7]. However, we work with rotation-based gates which allow computing a Boolean function by leaving the Boolean domain [12]. Therefore, this approach may be viewed as a step to explore synthesis of reversible functions by gates other than generalized Toffoli and Fredkin gates. We show that applying the proposed approach improves (1) circuit size for multiple-control Toffoli gates from exponential in [13, Lemma 7.1] to polynomial and from $48n^2 + O(n)$ [13, Lemma 7.6] to $2n^2 + O(n)$, (2) circuit depth for quantum carry-ripple adders by a constant factor compared to [14], and (3) circuit size for quantum multiplexers from $O(n^2)$ to $O(n \log^2 n)$.

The reminder of this paper is organized as follows. In Section 2, we touch upon necessary background in reversible and quantum circuits. Readers familiar with quantum circuits may ignore this section. Section 3 summarizes the previous work on quantum and reversible circuit synthesis. In Section 4, the proposed rotation-based technique is described. In Section 5, we provide an extension of the proposed synthesis algorithm to handle a more general logic functions, i.e., functions with binary inputs and arbitrary outputs. Synthesis of several function families are discussed in Section 6, and finally Section 7 concludes the paper. A partial version of this paper was presented in [15].

2 Basic Concepts

A quantum bit, *qubit*, can be realized by a physical system such as a photon, an electron or an ion. In this paper, we treat a qubit as a mathematical object which represents a quantum state with two basic states $|0\rangle$ and $|1\rangle$. A qubit can get any linear combination of its basic states, called *superposition*, as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ where α and β are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$.

Although a qubit can get any linear combination of its basic states, when a qubit is *measured*, its state collapses into the basis $|0\rangle$ and $|1\rangle$ with the probability of $|\alpha|^2$ and $|\beta|^2$, respectively. It is also common to denote the state of a single qubit by a 2×1 vector as $[\ \alpha \quad \beta\ ]^T$ in Hilbert space H where superscript T stands for the transpose of a vector. A quantum system which contains n qubits is often called a *quantum register* of size n. Accordingly, an n-qubit quantum register can be described by an element $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \ldots \otimes |\psi_n\rangle$ (simply $|\psi_1\psi_2 \cdots \psi_n\rangle$) in the tensor product Hilbert space $H = H_1 \otimes H_2 \otimes \cdots \otimes H_n$.

An n-qubit *quantum gate* performs a specific $2^n \times 2^n$ unitary operation on selected n qubits. A matrix U is unitary if $UU^\dagger = I$ where $U^\dagger$ is the conjugate transpose of U and I is the identity matrix. The unitary matrix implemented by several gates acting on different qubits independently can be calculated by the tensor product of their matrices. Two or more quantum gates can be cascaded to construct a *quantum circuit*. For a set of k gates $g_1, g_2, \cdots, g_k$ cascaded in a quantum circuit C in sequence, the matrix of C can be calculated as $M_k M_{k-1} \cdots M_1$ where M_i is the matrix of the i^{th} gate ($1 \leq i \leq k$). For a quantum circuit

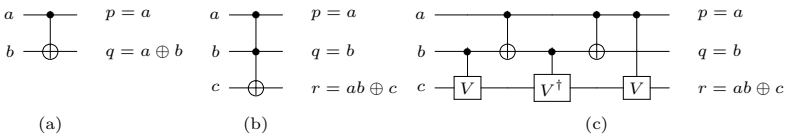

Fig. 1. CNOT (a) and Toffoli (b) gates. Decomposition of a Toffoli gate into 2-qubit gates (c) where $V = (1-i)(I+iX)/2$ [13].

with unitary matrix U and input vector ψ_1, the output vector is $\psi_2 = U\psi_1$.

Various quantum gates with different functionalities have been introduced. The θ-rotation gates ($0 \leq \theta \leq 2\pi$) around the x, y and z axes acting on one qubit are defined as Eq. (1). The single-qubit NOT gate is described by the matrix X in Eq. (2). The CNOT (controlled NOT) acts on two qubits (control and target) is described by the matrix representation shown in Eq. (2). The Hadamard gate, H, has the matrix representation shown in Eq. (2).

$$R_x(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, R_y(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, R_z(\theta) = \begin{pmatrix} e^{\frac{-i\theta}{2}} & 0 \\ 0 & e^{\frac{i\theta}{2}} \end{pmatrix} \quad (1)$$

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \text{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (2)$$

Given any unitary U over m qubits $|x_1 x_2 \cdots x_m\rangle$, a controlled-$U$ gate with k control qubits $|y_1 y_2 \cdots y_k\rangle$ may be defined as an $(m+k)$-qubit gate that applies U on $|x_1 x_2 \cdots x_m\rangle$ iff $|y_1 y_2 \cdots y_k\rangle$=$|11\cdots1\rangle$. For example, CNOT is the controlled-NOT with a single control, Toffoli is a NOT gate with two controls, and $CR_x(\theta)$ is a $R_x(\theta)$ gate with a single control. Similarly, a multiple-control Toffoli gate C^{k}NOT is a NOT gate with k controls. Fig. 1 shows CNOT and Toffoli gates. For a circuit implementing a unitary U, it is possible to implement a circuit for the controlled-U operation by replacing every gate by a controlled gate. In circuit diagrams, $\bullet$ is used for conditioning on the qubit being set to value one.

3 Previous Work

Synthesis of 0-1 unitary matrices, also called *permutation*, has been followed by several researchers during the recent years. Here, we review the recent approaches with favorable results. More information can be found in [7]. *Transformation-based* methods [16] iteratively select a gate to make a given function more similar to the identity function. These methods construct compact circuits mainly for permutations with repeating patterns in output codewords. *Search-based* methods [17] explore a search tree to find a realization. These methods are highly useful if the number of circuit lines and the number of gates in the final circuit are small. *Cycle-based* methods [18] decompose a given permutation into a set of disjoint (often small) *cycles* and synthesize individual cycles separately. These methods are mainly efficient for permutations without repeating output patterns. *BDD-based* methods [19] use binary decision diagrams to improve sharing between controls of reversible gates. These techniques scale better than others. However, they require a large number of ancilla qubits.

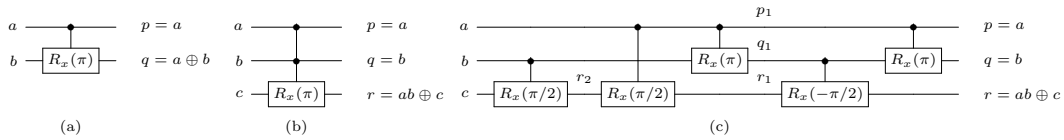

Fig. 2. New definitions for CNOT (a) and Toffoli (b) gates using controlled rotation gates. Decomposition of a Toffoli gate into 5 2-qubit controlled-rotation gates (c).

Quantum-logic synthesis deals with general unitary matrices and is more challenging than reversible-logic synthesis. Synthesis of an arbitrary unitary matrix from a universal set of gates including one-qubit operations and CNOTs has a rich history. Barenco et al. in 1995 [13] showed that the number of CNOT gates required to implement an arbitrary unitary matrix over n qubits was $O(n^3 4^n)$. As of 2012, the most compact circuit constructions use $\frac{23}{48}4^n - \frac{3}{2}2^n + \frac{4}{3}$ CNOTs [8, 20] and $\frac{1}{2}4^n + \frac{1}{2}2^n - n - 1$ one-qubit gates [21]. The sharpest lower bound on the number of CNOT gates is $\lceil \frac{1}{4}(4^n - 3n - 1) \rceil$ [22]. Different trade-offs between the number of one-qubit gates and CNOTs are explored in [23].

4 Rotation-Based Synthesis of Boolean Functions

In this section, we address the problem of automatically synthesizing a given Boolean function f by using rotation and controlled-rotation gates around the x axis. In this paper, we change the basis states as $\hat{0} = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$ and $\hat{1} = R_x(\pi)\hat{0} = \begin{bmatrix} 0 & -i \end{bmatrix}^T$. With this definition of $\hat{0}$ and $\hat{1}$, the basis states remain orthogonal. Also, inversion (i.e., the NOT gate) from one basis state to the other is simply obtained by a $R_x(\pi)$ gate.[1] Subsequently, the CNOT gate can be described by using the $CR_x(\pi)$ operator shown in Fig. 2(a). In addition, the Toffoli gate may be described by using the $C^2R_x(\pi)$ operator illustrated in Fig. 2(b). Toffoli gate can be implemented using 5 controlled-rotation operators as demonstrated in Fig. 2(c). Recall that a 3-qubit Toffoli gate needs 5 2-qubit gates if $|0\rangle$ and $|1\rangle$ are used as the basis states (Fig. 1(c)).

For a 2-qubit $CR_x(\theta)$ gate with a control qubit a and a target qubit b, the first output is equal to a. However, the second output depends on both the control line a and the target line b. We use the notation $aR_x(\theta)b$ to describe the second output. Furthermore, we write $R_x(\theta)b$ to unconditionally apply a single-qubit $R_x(\theta)$ to the qubit b. Additionally, one can show that for binary variables a, b, c we have $aR_x(\theta_1)[aR_x(\theta_2)b] = aR_x(\theta_1 + \theta_2)b$, $aR_x(\theta_1)\,[bR_x(\theta_2)c] = bR_x(\theta_2)\,[aR_x(\theta_1)c]$, $aR_x(\pi)\hat{1} = \sim a$ ($\sim$ is used for negation), and $aR_x(\pi)\hat{0} = a$.

Definition 1 $\hat{0}$ and all variables are in the *rotation-based factored* (factored in short) form. If h and g are in the factored form, then $R_x(\theta)h$ and $gR_x(\theta)h$ are in the factored form too.

In a quantum circuit synthesized with $R_x(\theta)$ and $CR_x(\theta)$ operators, all outputs and intermediate signals in the given circuit can be described in the factored form. For example, the

[1] While we used $\hat{0}$ and $\hat{1}$ as the basis states, the presented algorithm can be easily modified to be applicable to quantum functions described in terms of $|0\rangle$ and $|1\rangle$. An alternate solution is to define the following operators and use M to transform the $|0\rangle$ and $|1\rangle$ states to $\hat{0}$ and $\hat{1}$ states and operator M^{-1} to perform the reverse transformation. Hence to compute in $|0\rangle$ and $|1\rangle$ basis, one needs to apply M and M^{-1} single-qubit operators before and after the computation done in the $\hat{0}$ and $\hat{1}$ basis, respectively. Notice that M and M^{-1} are rotations around the z axis.

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

output r in Fig. 2(c) can be described as $r = [aR_x(\pi)b]\, R_x(-\pi/2)\, [aR_x(\pi/2)\, [bR_x(\pi/2)c]]$.

Definition 2 $\hat{0}$ and all variables are *rotation-based cascade* (cascade in short) expressions. If h is a cascade expression and v is a variable $\notin h$, then $R_x(\theta)h$ and $vR_x(\theta)h$ are cascade expressions too ($\forall\theta$).

A cascade expression can be expressed as $R_x(\theta_0)\left[v_1R_x(\theta_1)\left[v_2R_x(\theta_2)\cdots\left[v_nR_x(\theta_n)\hat{0}\right]\,\right]\,\right]$. The problem of realizing a function with $R_x(\theta)$ and $CR_x(\theta)$ operators is equivalent to finding a cascade expression for the function. To do this, we first introduce a graph-based data structure in the form of a decision diagram for representing functions.

4.1 A Rotation-Based Data Structure

The concept of binary decision diagram (BDD) was first proposed by Lee [24] and later developed by Akers [25] and then by Bryant [26], who introduced Reduced Ordered BDD (ROBDD) and proved its canonicity property. Bryant also provided a set of operators for manipulating ROBDDs. In this paper, we omit the prefix RO. BDD has been extensively used in classical logic synthesis. Furthermore, several variants of BDD were also proposed for logic synthesis [19], verification [27, 28, 29] and simulation [30, 31] of reversible and quantum circuits. In this section, we describe a new decision diagram for the representation of functions based on rotation operators. Next, we use it to propose a synthesis framework for logic synthesis with rotation gates.

Definition 3 A Rotation-based Decision Diagram (RbDD) is a directed acyclic graph with three types of nodes: a single terminal node with value $\hat{0}$, a weighted root node, and a set of non-terminal (internal) nodes. Each internal node represents a function and is associated with a binary decision variable with two outgoing edges: a weighted $\hat{1}$-edge (solid line) leading to another node, the $\hat{1}$-child, and a non-weighted $\hat{0}$-edge (dashed line) leading to another node, the $\hat{0}$-child. The weights of the root node and $\hat{1}$-edges are in the form of $R_x(\theta)$ matrices. We assume that $-\pi < \theta \leq \pi$. When a weight (either for an edge or the root node) is the identity matrix (i.e., $R_x(0) = I$), it is not shown in the diagram.

The left RbDD in Fig. 3(a) shows an internal node f with decision variable a, the corresponding $\hat{0}$ and $\hat{1}$ edges, and child nodes f_0 and f_1. The relation between the RbDD nodes in this figure is as follows. If $a = \hat{1}$, then $f = R_x(\theta)f_1$ else $f = f_0$. In addition, if f is a weighted root node as shown in the right RbDD in Fig. 3(a), then for $a = \hat{1}$ we have $f = R_x(\theta_r)R_x(\theta)f_1 = R_x(\theta_r + \theta)f_1$; otherwise $f = R_x(\theta_r)f_0$. Similar to BDDs, in RbDDs isomorphic sub-graphs which are nodes with the same functions are merged. Additionally, if the $\hat{0}$-child and the $\hat{1}$-child of a node are the same and the weight of $\hat{1}$-edge is $R_x(0) = I$, then that node is eliminated. Using these two reduction rules and a given total ordering $\prec$ on input variables, one can uniquely construct the RbDD of a given function. Notably, a decision diagram called DDMF was proposed in [28], where each edge can represent any unitary matrix including rotation operators. DDMF was used for verification of quantum circuits.

For a given function f with n binary variables $v_1, v_2, \cdots, v_n$, each value assignment to $v_1, v_2, \cdots, v_n$ corresponds to a path from the root to the terminal node in the RbDD of f. Assuming the variable ordering $v_1 < v_2 < \cdots < v_n$, the corresponding path can be identified by a top-down traversal of the RbDD starting from the root node. For each node visited during the traversal, we select the edge corresponding to the value of its decision variable v_i. Denote the weight of the root node by w_0 and the weight of the selected edges by

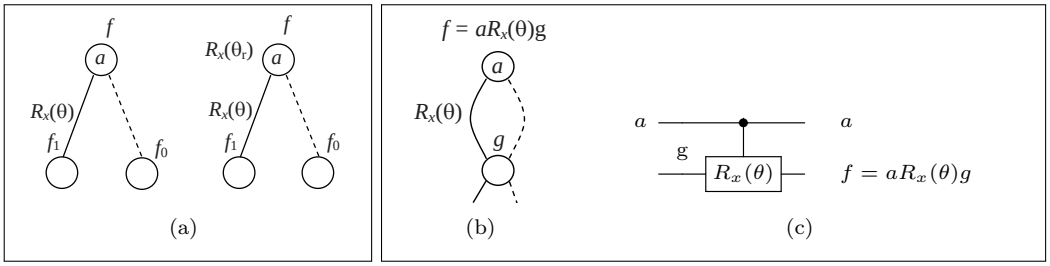

Fig. 3. (a) Internal structure of a rotation-based decision diagram (RbDD) without and with a weighted root. (b) For a node f, if the $\hat{0}$-child and the $\hat{1}$-child are the same node g, f can be directly realized by a $R_x(\theta)$ operator as shown in (c).

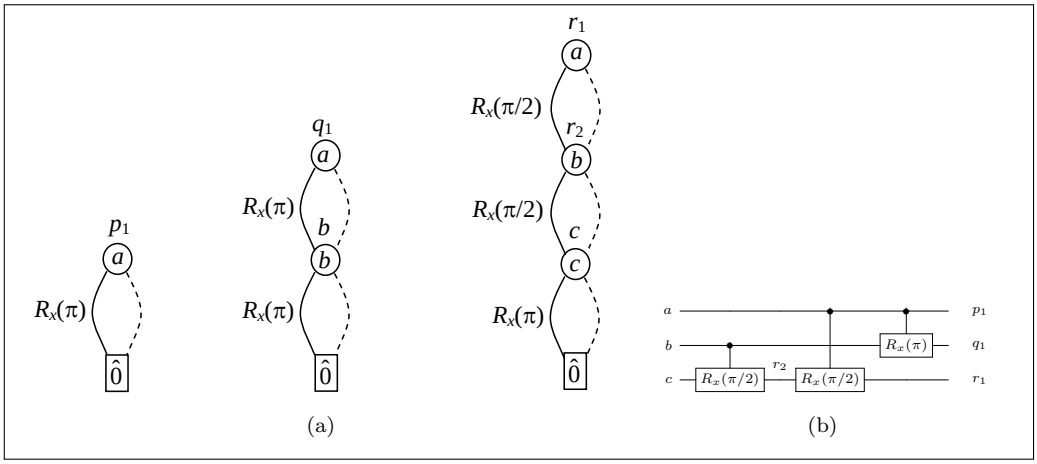

Fig. 4. RbDDs for intermediate signals of a 3-input Toffoli gate shown in Fig. 2(c), redrawn in (b). In this figure, we have $q_1 = aRx(\pi)b$, $r_1 = aR_x(\pi/2)r_2$, and $r_2 = bR_x(\pi/2)c$.

$w_1, w_2, \cdots, w_{n-1}$. We have $f(v_1, v_2, \cdots, v_n) = w_0 w_1 \cdots w_{n-1}\hat{0} = w_0 w_1 \cdots w_{n-1} \begin{bmatrix} 1 & 0 \end{bmatrix}^T$. If a $\hat{0}$-edge is selected for variable v_i (i.e., if $v_i = \hat{0}$), we have $w_i = I$. Note that when the $\hat{0}$-child and the $\hat{1}$-child of a node f are the same node g, then that node can be directly realized by a $R_x(\theta)$ operator, as $f = aR_x(\theta)g$ demonstrated in Fig. 3(b) and Fig. 3(c), in terms of its child. Fig. 4(a) shows the RbDDs of functions p_1, q_1 and r_1 in Fig. 2(c) (reproduced in Fig. 4(b)). Every RbDD with a chain structure such as the ones shown in Fig. 4(a) is associated with a cascade expression and can be realized with rotation and controlled-rotation operators.

Suppose that the RbDD for a function f is given. The RbDD for $h = R_x(\gamma)f$ can be obtained by multiplying the root weight of f by $R_x(\gamma)$. To obtain $h = fR_x(\gamma)g$ for given RbDDs of f and g, we use the `apply` operator.[2] In this context, f and g are called RbDD operands of h. The `apply` operator is implemented by a recursive traversal of the two RbDD operands. For each pair of nodes in f and g visited during the traversal, an internal node is added to the resulting RbDD by utilizing the following rules which depend on the selected variable ordering $\prec$ (also see Fig. 5). We assume that f and g have two general RbDDs

[2]In general, for a binary operation `op` and two BDDs of functions f and g, the `apply` operator computes a BDD for f `op` g [26].

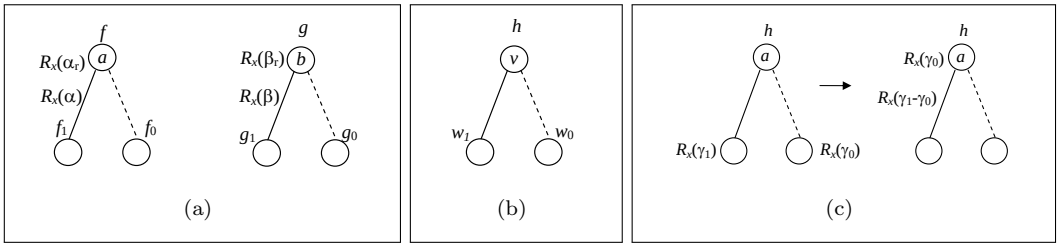

Fig. 5. (a) Operands for operation $h = fR_x(\gamma)g$. (b) The result of `apply` operator which adds a new node to the resulting RbDD h by using one of the three rules: if $a < b$ (Rule 1), $v = a$, $w_1 = [R_x(\alpha_r + \alpha)f_1]R_x(\gamma)g$, $w_0 = [R_x(\alpha_r)f_0]R_x(\gamma)g$. If $b < a$ (Rule 2), $v = b$, $w_1 = fR_x(\gamma)[R_x(\beta_r + \beta)g_1]$, $w_0 = fR_x(\gamma)[R_x(\beta_r)g_0]$. If $a = b$ (Rule 3), $v = a = b$, $w_1 = [R_x(\alpha_r + \alpha)f_1]R_x(\gamma)[R_x(\beta_r + \beta)g_1]$, $w_0 = [Rx(\alpha_r)f_0]R_x(\gamma)[R_x(\beta_r)g_0]$. (c) Weight modification for the `apply` operator to maintain canonicity of the resulting RbDD.

shown in Fig. 5(a). The `apply` operator is recursively called with the terminal conditions $\hat{0}R_x(\theta)v = v$ and $\hat{1}R_x(\theta)v = R_x(\theta)v$.

- **Rule 1** ($a < b$) The new node for h is a. The weights of $\hat{1}$-child and $\hat{0}$-child are $[R_x(\alpha_r + \alpha)f_1]R_x(\gamma)g$, and $[R_x(\alpha_r)f_0]R_x(\gamma)g$, respectively.
- **Rule 2** ($b < a$) The new node for h is b. The weights of $\hat{1}$-child and $\hat{0}$-child are $fR_x(\gamma)[R_x(\beta_r + \beta)g_1]$, and $fR_x(\gamma)[R_x(\beta_r)g_0]$, respectively.
- **Rule 3** ($a = b$) The new node for h is a (or b). The weights of $\hat{1}$-child and $\hat{0}$-child are $[R_x(\alpha_r + \alpha)f_1]R_x(\gamma)[R_x(\beta_r + \beta)g_1]$, and $[Rx(\alpha_r)f_0]R_x(\gamma)[R_x(\beta_r)g_0]$, respectively.

After recursive computation of the $\hat{1}$-child and $\hat{0}$-child of h, to maintain the canonicity of the resulting RbDD, isomorphic sub-graphs are merged and if the $\hat{0}$-child and the $\hat{1}$-child of a node are the same and the weight of the $\hat{1}$-edge is $R_x(0) = I$, then that node will be eliminated. In addition, to make RbDD of h canonical, the resulting weights for the $\hat{1}$-child and the $\hat{0}$-child of h should be modified by the method illustrated in Fig. 5(c). Fig. 6(a) demonstrates the result of performing `apply` operator on q_1 and r_1 in Fig. 4(a), redrawn in Fig. 6(a), to obtain $r = q_1R_x(-\pi/2)r_1$. To construct RbDD for r, one needs to initially apply Rule 3 because both q_1 and r_1 use a as roots. Accordingly, $w_1 = [R_x(\pi)b]R_x(-\pi/2)[R_x(\pi/2)r_2]$ and $w_0 = bR_x(-\pi/2)r_2$. To continue, consider w_1 and note that both $[R_x(\pi)b]$ and $[R_x(\pi/2)r_2]$ use b.[3] As a result, applying Rule 3 leads to $w_{1,1} = [R_x(0)\hat{0}]R_x(-\pi/2)[R_x(\pi/2 + \pi/2)c]$ and $w_{1,0} = [R_x(\pi)\hat{0}]R_x(-\pi/2)[R_x(\pi/2)c]$. On the other hand, applying Rule 3 on w_0 leads to $w_{0,1} = [R_x(\pi)\hat{0}]R_x(-\pi/2)[R_x(\pi/2)c]$ and $w_{0,0} = [R_x(0)\hat{0}]R_x(-\pi/2)c$. Using terminal conditions results in $w_{1,1} = R_x(\pi)c$, $w_{1,0} = c$, $w_{0,1} = c$, and $w_{0,0} = c$. Since $w_{0,1} = w_{0,0} = c$, we can remove variable b as the $\hat{0}$-child of a. The final figure in Fig. 6(a) is obtained after eliminating redundant nodes and edges.[4]

[3] To understand the RbDDs of $[R_x(\pi)b]$ and $[R_x(\pi/2)r_2]$, recall RbDDs of b and r_2 in Fig. 6(a) and use weights $R_x(\pi)$ and $R_x(\pi/2)$ for roots of b and r_2, respectively.

[4] Note that the commutative property of matrix multiplication for $R_x(\theta)$ matrices is critical for the `apply` operator. Performing `apply` as described may not generate the correct result for decision diagrams with non-commutative weights.

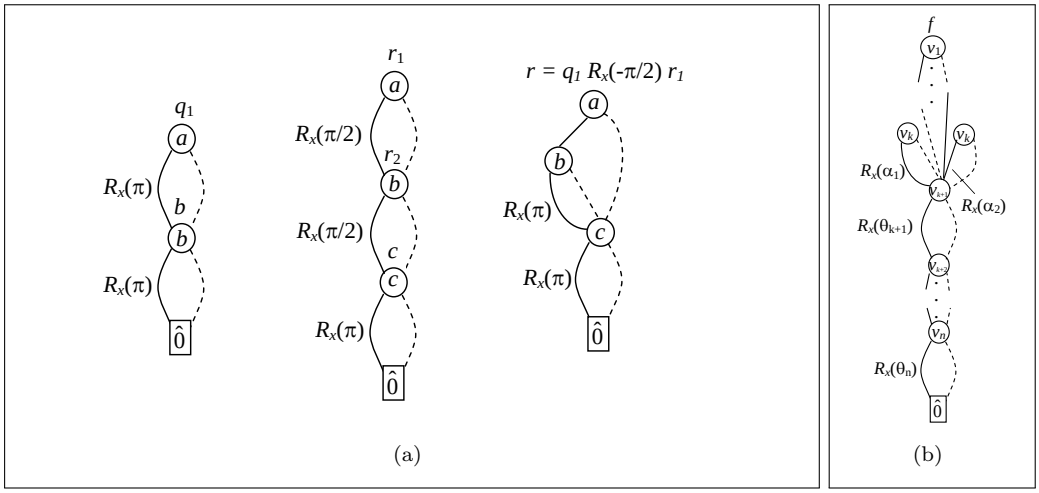

Fig. 6. (a) An example of performing `apply` operator on two RbDDs. In the first call of the `apply` operator, $w_1 = [R_x(\pi)b]R_x(-\pi/2)[R_x(\pi/2)r_2]$ and $w_0 = bR_x(-\pi/2)r_2$. The final figure is obtained after eliminating redundant nodes and edges. (b) A general RbDD structure with r-linear variables $v_{k+1}, \cdots, v_n$.

4.2 Functional Decomposition and r-Linearity

The problem of realizing a function f using $R_x(\theta)$ and $CR_x(\theta)$ operators is equivalent to finding a rotation-based factored form for f, which can be performed by recursive bi-decomposition of f.

Definition 4 *Rotation-based bi-decomposition* (bi-decomposition in short) of f is defined as finding functions g and h and value γ such that $f = gR_x(\gamma)h$.

We use bi-composition of a given function f to construct $f = gR_x(\gamma)h$. Subsequently, g and h are recursively bi-decomposed, which will eventually result in a factored form of f. The bi-decomposition algorithm is based on the notion of r-linearity.

Definition 5 For function $f(v_1, \cdots, v_{i-1}, v_i, v_{i+1}, \cdots, v_n)$, variable v_i is *r-linear* if there exists a rotation value θ_i such that for every value assignment to $v_1, \cdots, v_{i-1}, v_{i+1}, \cdots, v_n : f_{v_i} = R_x(\theta_i)f_{\overline{v}_i}$, where $f_{v_i} = f(v_1, \cdots, v_{i-1}, \hat{1}, v_{i+1}, \cdots, v_n)$ and $f_{\overline{v}_i} = f(v_1, \cdots, v_{i-1}, \hat{0}, v_{i+1}, \cdots, v_n)$. A variable is *r-nonlinear* if it is not r-linear.

Now we present a number of key results.

Lemma 1 Consider a function $f(v_1, v_2, \cdots, v_n)$ with variable ordering $v_1 < v_2 < \cdots < v_n$ and assume that $k + 1 \le i \le n$. Iff each variable v_i is r-linear, then there is only one RbDD node n_i for each r-linear decision variable v_i. The weight of the $\hat{1}$-edge of n_i is $R_x(\theta_i)$.

Proof. The proof is by induction on $v_n, v_{n-1}, v_{n-2}, \cdots, v_{k+1}$ starting from v_n. ∎

Let v_k be the lowest indexed r-nonlinear variable after which $v_{k+1}, v_{k+2}, \cdots, v_n$ are r-linear variables of f. From Lemma 1, for $k + 1 \le j \le n$ we have $f_{v_j} = R_x(\theta_j)f_{\overline{v}_j}$ where θ_j is fixed independent of $v_1, v_2, \cdots, v_{j-1}, v_{j+1}, \cdots, v_n$ values. As illustrated in Fig. 6(b), every path from the root node of the RbDD to the terminal node will either go through an internal node with decision variable v_k or it will skip any such node and directly go the single RbDD node with decision variable v_{k+1}. For the latter case, $f_{v_k} = R_x(0)f_{\overline{v}_k} = f_{\overline{v}_k}$ and for any former case $f_{v_k} = R_x(\alpha_i)f_{\overline{v}_k}$ for some (vs. all) $v_1, \cdots, v_{k-1}, v_{k+1}, \cdots, v_n$. Additionally, the number

Algorithm 1 `factor` (f)

1: If all variables are r-linear, then return the corresponding cascade expression for f.
2: Find the lowest indexed r-nonlinear variable v_k after which $v_{k+1}, v_{k+2}, \cdots, v_n$ are r-linear.
3: Bi-decompose f using v_k as $f = g_1 R_x(\gamma) h$ where g_1, h, and γ are given in Theorem 1.
4: Return [`factor(`g_1`)`]$R_x(\gamma)$[`factor(`h`)`].

of different rotation angles (e.g., α_1, α_2 in Fig. 6(b)) for variable v_k is equal to the number of internal nodes with decision variable v_k in the RbDD.

Definition 6 The *degree of r-nonlinearity* of variable v_k, r-deg(v_i), is $m-1$ where m is the number of different rotation angles α_i (including 0 if any) that $f_{v_k} = R_x(\alpha_i) f_{\overline{v}_k}$ for some $v_1, \cdots, v_{k-1}, v_{k+1}, \cdots, v_n$. For r-linear variables the degree of r-nonlinearity is zero.

As an example, consider the RbDD of r in Fig. 6(a) and note that r-deg$(b) = 1$ as there are two rotation angles (i.e., 0 and π) for b. Similarly, r-deg$(c) = 0$ and c is r-linear.

Lemma 2 Let m denote the number of internal nodes with decision variable v_k. If all paths from the root node of the RbDD to the terminal node go through an internal node with decision variable v_k, r-deq$(v_k) = m - 1$; otherwise r-deg$(v_k) = m$.

Proof. The proof follows from considering the general structure of RbDDs and the definition of r-nonlinearity. ∎

Theorem 1 Consider a function $f(v_1, v_2, \cdots, v_n)$ with variable ordering $v_1 < v_2 < \cdots < v_n$. Define g such that if $f_{v_k} = R_x(\alpha_1) f_{\overline{v}_k}$ then $g = \hat{1}$; otherwise $g = \hat{0}$. Assume that $v_{k+1}, v_{k+2}, \cdots, v_n$ are r-linear variables of f and v_k is a r-nonlinear variable of f with r-deg$(v_k) = m-1$. Additionally, for each value assignment to variables $v_1, \cdots, v_{k-1}, v_{k+1}, \cdots, v_n$ suppose exactly one of the following m relations holds: $f_{v_k} = R_x(\alpha_1) f_{\overline{v}_k}$, $f_{v_k} = R_x(\alpha_2) f_{\overline{v}_k}$, $\cdots$, $f_{v_k} = R_x(\alpha_m) f_{\overline{v}_k}$. We have

I f can be bi-decomposed as $f = g_1 R_x(\gamma) h$ where $g_1 = v_k R_x(\pi) g$, $\gamma = (\alpha_2 - \alpha_1)/2$, $h = g_1 R_x(-\gamma) f$.

II g_1 is a function of $v_1, v_2, \cdots, v_k$, i.e., g_1 is invariant with respect to $v_{k+1}, v_{k+2}, \cdots, v_n$.

III v_k is a r-linear variable of g_1.

IV h is a function of $v_1, v_2, \cdots, v_n$ and $v_{k+1}, v_{k+2}, \cdots, v_n$ are r-linear variables of h.

V r-deg(v_k) in h is $\leq m - 2$.

Proof. We initially prove that function g is invariant with respect to $v_{k+1}, v_{k+2}, \cdots, v_n$, i.e., $g_{v_i} = g_{\overline{v}_i}$ for $k+1 \leq i \leq n$. Since v_i is r-linear, there exists θ_i such that for all $v_1, \cdots, v_{i-1}, v_{i+1}, \cdots, v_n$ values, $f_{v_i} = R_x(\theta_i) f_{\overline{v}_i}$ which results in $f_{v_i v_k} = R_x(\theta_i) f_{\overline{v}_i v_k}$ and $f_{v_i \overline{v}_k} = R_x(\theta_i) f_{\overline{v}_i \overline{v}_k}$. From the definition of g we have:

- If $f_{v_i v_k} = R_x(\alpha_1) f_{v_i \overline{v}_k}$, then $g_{v_i} = \hat{1}$, else $g_{v_i} = \hat{0}$.
- If $f_{\overline{v}_i v_k} = R_x(\alpha_1) f_{\overline{v}_i \overline{v}_k}$, then $g_{\overline{v}_i} = \hat{1}$, else $g_{\overline{v}_i} = \hat{0}$.

Combining these relations proves $g_{v_i} = g_{\overline{v}_i}$:

$$f_{v_i v_k} = R_x(\alpha_1) f_{v_i \overline{v}_k} \Leftrightarrow R_x(\theta_i) f_{\overline{v}_i v_k} = R_x(\alpha_1 + \theta_i) f_{\overline{v}_i \overline{v}_k} \Leftrightarrow f_{\overline{v}_i v_k} = R_x(\alpha_1) f_{\overline{v}_i \overline{v}_k}$$

Since $g_1 = v_k R_x(\pi) g$, g_1 is also invariant with respect to $v_{k+1}, v_{k+2}, \cdots, v_n$ (part II). Moreover $g_{1v_k} = R_x(\pi) g$ and $g_{1\overline{v}_k} = g$ which results in $g_{1v_k} = R_x(\pi) g_{1\overline{v}_k}$, i.e., v_k in g_1 is r-linear (part III).

The first sentence of part IV is clear from the definition of $h = g_1 R_x(-\gamma) f$. As for the second one, note that $v_{k+1}, v_{k+2}, \cdots, v_n$ are r-linear variables of f. Additionally, g is invariant with respect to $v_{k+1}, v_{k+2}, \cdots, v_n$. Putting these facts together proves part IV.

Now we prove r-deg$(v_k) \leq m-2$ in $h = g_1 R_x(-\gamma) f$. For each value assignment to variables $v_1, v_2, \cdots, v_{k-1}, v_{k+1}, \cdots, v_n$ exactly one of the following m relations holds: $f_{v_k} = R_x(\alpha_1) f_{\overline{v}_k}$, $f_{v_k} = R_x(\alpha_2) f_{\overline{v}_k}$, $\cdots$, $f_{v_k} = R_x(\alpha_m) f_{\overline{v}_k}$. For each of the above cases, we examine the relation between h_{v_k} and $h_{\overline{v}_k}$:

- $f_{v_k} = R_x(\alpha_1) f_{\overline{v}_k}$: By definition $g = \hat{1}$ and we have:

  $h_{\overline{v}_k} = \hat{1} R_x(-\gamma) f_{\overline{v}_k} = R_x(-\gamma) f_{\overline{v}_k} \Rightarrow f_{\overline{v}_k} = R_x(\gamma) h_{\overline{v}_k}$
  $h_{v_k} = [R_x(\pi)\hat{1}] R_x(-\gamma) f_{v_k} = f_{v_k} = R_x(\alpha_1) f_{\overline{v}_k} = R_x(\alpha_1 + \gamma) h_{\overline{v}_k}, \gamma = (\alpha_2 - \alpha_1)/2 \Rightarrow$
  $h_{v_k} = R_x(\frac{\alpha_1+\alpha_2}{2}) h_{\overline{v}_k}$

- $f_{v_k} = R_x(\alpha_2) f_{\overline{v}_k}$: By definition $g = \hat{0}$ and we have:

  $h_{\overline{v}_k} = \hat{0} R_x(-\gamma) f_{\overline{v}_k} = f_{\overline{v}_k}$
  $h_{v_k} = [R_x(\pi)\hat{0}] R_x(-\gamma) f_{v_k} = R_x(-\gamma) f_{v_k} = R_x(-\gamma + \alpha_2) f_{\overline{v}_k}, \gamma = (\alpha_2 - \alpha_1)/2 \Rightarrow$
  $h_{v_k} = R_x(\frac{\alpha_1+\alpha_2}{2}) h_{\overline{v}_k}$

- $f_{v_k} = R_x(\alpha_i) f_{\overline{v}_k}$, $3 \leq i \leq m$: By definition $g = \hat{0}$ and $h_{v_k} = R_x(-\gamma + \alpha_i) h_{\overline{v}_k}$.

The first two cases result in the same relation between h_{v_k} and $h_{\overline{v}_k}$ as $h_{v_k} = R_x(\frac{\alpha_1+\alpha_2}{2}) h_{\overline{v}_k}$. The remaining $m-2$ cases result in at most $m-2$ different relations between h_{v_k} and $h_{\overline{v}_k}$. Therefore, the total number of different relations between h_{v_k} and $h_{\overline{v}_k}$ is $\leq m-1$. Accordingly, r-deg(v_k) in h is $\leq m-2$ (part V).

Finally, from $h = g_1 R_x(-\gamma) f$ it follows that $g_1 R_x(\gamma) h = g_1 R_x(\gamma) [g_1 R_x(-\gamma) f]$. Consider $g_1 = v_k R_x(\pi) g$ and assume $g = \hat{1}$ (or $g = \hat{0}$) which leads to $g_1 = v_k R_x(\pi)\hat{1} = \sim v_k$ (or $g_1 = v_k R_x(\pi)\hat{0} = v_k$). Altogether for both $v_k = \hat{1}$ and $v_k = \hat{0}$, we have $g_1 R_x(\gamma) [g_1 R_x(-\gamma) f] = f$. Hence, f can be bi-decomposed as $f = g_1 R_x(\gamma) h$ (part I). ∎

Using the proposed bi-decomposition approach, f can be bi-decomposed into $f = g_1 R_x(\gamma) h$ where g_1 and h are themselves recursively bi-decomposed until a rotation-based factored form is obtained.

Theorem 2 The proposed bi-decomposition approach always results in a cascade expression for a given function f.

Proof. Following the definitions given in Theorem 1 for $f = g_1 R_x(\gamma) h$, since g_1 is invariant of $v_{k+1}, v_{k+2}, \cdots, v_n$ and v_k in g is r-linear and r-deg(v_k) in h is $\leq m-2$, the recursion will finally stop at terminal cases where g_1 and/or h have directly realizable RbDDs — all variables

Algorithm 2 g_1-`factor(RbDD`$_f$`)`

1: Change all of the weights to $R_x(0) = I$.
2: Create a RbDD node v_k with $w_1 = R_x(\pi)$ and $w_0 = I$ to the terminal node (i.e., $\hat{0}$).
3: Redirect all edges toward n_1 to node v_k and make the weight of all such edges $R_x(\pi)$.
4: Redirect all edges toward $n_2, n_3, \cdots, n_m$ to node v_k and make the weight of all such edges $R_x(0)$.
5: Discard nodes $n_2, n_3, \cdots, n_m$.
6: Merge isomorphic sub-graphs, eliminate nodes with the same $\hat{0}$-child and $\hat{1}$-child exactly if the weight of the $\hat{1}$-edge is $R_x(0) = I$. Update weights of the RbDD to make the RbDD of g_1 canonical.

are r-linear in the functions and they have rotation-based cascade expressions corresponding to RbDDs with a chain structure.[5] ∎

Algorithm 1 uses the proposed recursive bi-decomposition approach to generate a rotation-based factored form for a given function f. All steps in Algorithm 1 can be directly performed on RbDDs. If the RbDD of a function f is a chain structure, we have a cascade expression for f (Step 1). For Step 2 as depicted in Fig. 6(b) and according to Lemma 1, identifying v_k is equivalent to identifying the lower chain-structure part of the RbDD. As for Step 3, according to Lemma 2 values $\alpha_1, \alpha_2, \cdots, \alpha_m$ can be obtained from weights of the $\hat{1}$-edges of nodes with decision variables v_k. Hence, $\gamma = (\alpha_2 - \alpha_1)/2$ is obtained. Let $n_i (1 \leq i \leq m)$ denote nodes with decision variable v_k and $\hat{1}$-edges weight $R_x(\alpha_i)$. Starting from the RbDD of f, one can perform Algorithm 2 to construct RbDD of g_1. Having the RbDDs for g_1 and f, the RbDD of $h = g_1 R_x(-\gamma) f$ can be obtained by using the `apply` operation. As an example of Algorithm 2, see RbDDs of s and g_1 in Fig. 8 and Fig. 9 where $v_k = c$ and $m = 2$. This example is described in detail in Section 6.

The final form after `apply` is $f = g_1 R_x(\gamma_1) \left[g_2 R_x(\gamma_2) \left[g_3 R_x(\gamma_3) \cdots \left[g_k R_x(\gamma_k) \hat{0} \right] \right] \right]$. Note that g_i functions should also be decomposed. The `factor` algorithm is not optimal. In particular, f can be rewritten as $f = g_{p_1} R_x(\gamma_{p_1}) \left[g_{p_2} R_x(\gamma_{p_2}) \left[g_{p_3} R_x(\gamma_{p_3}) \cdots \left[g_{p_k} R_x(\gamma_{p_k}) \hat{0} \right] \right] \right]$ where $(p_1, p_2, \cdots, p_k)$ is a permutation of $1, 2, \cdots, k$. Different permutations of $1, 2, \cdots, k$ may result in different number of gates after synthesis. For example, consider the RbDDs of the output s in a 4-input Toffoli gate, shown in Fig. 8, for two different variable orderings $a > b > c > d$ and $a > b > d > c$. In Fig. 8(b), d is r-linear. However, none of the variables in Fig. 8(c) are r-linear. Accordingly, the proposed approach results in fewer gates for $a > b > c > d$. The former case is further discussed in Section 6. Indeed, working with $a > b > d > c$ leads to $s = g_1 R_x(-\pi/2) h$ where $g_1 = abd \oplus c$ is a 4-input Toffoli gate targeted on the last qubit c for $a > b > d > c$.

5 Working with Arbitrary Outputs

For the input vector U, a function f with binary inputs and outputs can be written as $f(U) = \hat{g}_1(U) R_x(\gamma_1) \left[\hat{g}_2(U) R_x(\gamma_2) \left[\cdots \left[\hat{g}_k(U) R_x(\gamma_k) \hat{0} \right] \right] \right]$. Since functions $\hat{g}_i(U)$ only take values $\hat{0}$ and $\hat{1}$, $f(U)$ can also be represented as $f(U) = R_x(g_1(U)\gamma_1 + g_2(U)\gamma_2 + \cdots + g_k(U)\gamma_k)\hat{0}$

[5] As a result of Lemma 1, in a function with chain structured RbDD, all variables are r-linear.

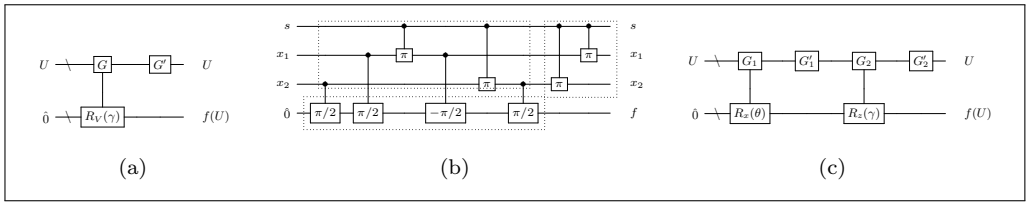

Fig. 7. (a) Quantum circuit performing $f(U) = R_x(\gamma(U))\hat{0}$. (b) Quantum 3-input multiplexer. Dashed boxes represent G, G', and $R_x(\gamma)$ in (a). Only rotation angles are reported for $R_x(\theta)$ gates. (c) Quantum circuit performing $f(U) = R_z(\gamma(U))R_x(\theta(U))\hat{0}$.

where $g_i(U)$ values are either zero (0) or one (1).[6] Define $\gamma(U) = g_1(U)\gamma_1 + g_2(U)\gamma_2 + \cdots + g_k(U)\gamma_k$ which leads to $f(U) = R_x(\gamma(U))\hat{0}$. Accordingly, the structure of the synthesized circuit can be represented as Fig. 7(a). In this figure, G is a circuit that constructs $\gamma(U)$ and G' is the inverse of G. Note that G' should be used only if one wants to keep input lines unchanged. To clarify the roles of G and G, see the 3-input multiplexer circuit $f = sx_1 + \bar{s}x_2$ synthesized by the `factor` algorithm in Fig. 7(b). If instead of $\hat{0}$, another quantum value q is used in this circuit as the initial value for the input, the resulting circuit implements $f(U) = R_x[\gamma(U)]\, q$. The constant ancilla register in Fig. 7(a) may not be necessary in some case. For example, the controlled rotation $R_x(\pi)$ with control qubit a and target $\hat{0}$ generates a as the second output and the use of the controlled rotation $R_x(\pi)$ in this case is unnecessary (i.e., $aR_x(\pi)\hat{0} = a$). Section 6 shows several examples.

Now consider a given function that for given basis input vectors generates a general value $f(U) = \begin{bmatrix} f_0(U) & f_1(U) \end{bmatrix}^T$. Since $|f_0(U)|^2 + |f_1(U)|^2 = 1$, we may rewrite $f(U)$ as:

$$f(U) = e^{i\delta(U)} \begin{bmatrix} e^{-i\gamma(U)/2} \cos\frac{\theta(U)}{2} & -ie^{i\gamma(U)/2} \sin\frac{\theta(U)}{2} \end{bmatrix}^T$$

Hence, $f(U)$ can be expressed as $f(U) = e^{i\delta(U)} R_z(\gamma(U))R_x(\theta(U))\hat{0}$ where R_z is the rotation operator around the z axis. We can ignore the global phase $e^{i\delta(U)}$ since it has no observable effects [1]. Therefore, one can effectively write $f(U) = R_z(\gamma(U))R_x(\theta(U))\hat{0}$. Note that $R_z(\gamma)R_x(\theta)\hat{0}$ results from θ rotation of $\hat{0}$ around the x axis followed by γ rotation around the z axis in the Bloch sphere. The quantum circuit for $f(U) = R_z(\gamma(U))R_x(\theta(U))\hat{0}$ can be synthesized as:

- Synthesize $g(U) = R_x(\theta(U))\hat{0}$ by using the `factor` algorithm.
- Synthesize $h(U) = R_z(\gamma(U))\hat{0}$ by using the `factor` algorithm.
- Cascade the resulting circuits as depicted in Fig. 7(c). In this figure, G_1 and G_2 are for $g(U)$ and $h(U)$, respectively. Accordingly, G_1' and G_2' are the inverse circuits of G_1 and G_2.

6 Results

Multiple-control Toffoli gate. Consider a **4-input Toffoli** gate in Fig. 8(a) and the RbDD of the target output in Fig. 8(b) with variable ordering $a < b < c < d$. Comparing the

[6] To prove, assign arbitrary values $\hat{0}$ and $\hat{1}$ to $\hat{g}_i$ terms and consider the resulting rotations by different γ_i values.

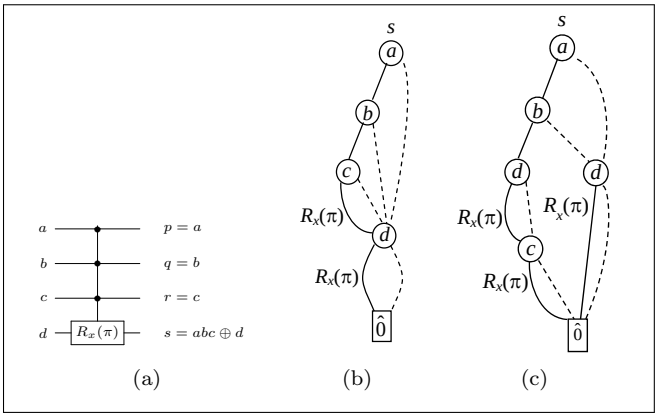

Fig. 8. 4-input Toffoli gate (a), the RbDD for the 4th output s in two cases: if $a > b > c > d$ (b), and if $a > b > d > c$ (c).

RbDD of s with the general RbDD structure in Fig. 6(b) reveals that variable c corresponds to v_k. Additionally, r-deg(c)=1, $\alpha_1 = 0$ and $\alpha_2 = \pi$ which result in $\gamma = \pi/2$ (Theorem 1).[7]

Function s can be bi-decomposed as $s = g_1 R_x(-\pi/2)h$ where $g_1 = cR_x(\pi)g$. RbDDs for g and g_1 are shown in Fig. 9(a) and Fig. 9(b), respectively.[8] Note that g_1 is a 3-input Toffoli gate (see RbDD of r in Fig. 6(a)), which can be synthesized as in Fig. 2(c). As for function h, it can be written as $h = g_1 R_x(\pi/2)s$. The RbDD for h (by the `apply` operator) is shown in Fig. 9(c). Subsequently, h can be bi-decomposed as $h = g_2 R_x(-\pi/4)h_1$ where $g_2 = aR_x(\pi)b$ (by algorithm 2) and $h_1 = g_2 R_x(\pi/4)h$ (by the `apply` operator). The resulting RbDDs for g_2 and h_1 are shown in Fig. 9(d) and Fig. 9(e). Finally, the factored form for s is $s = g_1 R_x(-\pi/2)[g_2 R_x(-\pi/4)h_1]$.

Due to the chain structure of g_2 and h_1, they may be directly realized by using controlled-rotation operators. Note that when realizing g_1, we also implement g_2. The final circuit is shown in Fig. 10. The first subcircuit generates output s whereas the remaining gates generate outputs p, q and r.

As a direct extension of the above approach, consider a **multiple-control Toffoli gate** on $n+1$ qubits with controls $i_1, i_2, ..., i_n$ and target j. Toffoli output can be written as $j = i_1 i_2 \cdots i_n \oplus j$. Assume $i_1 < i_2 < \cdots < i_n < j$. It can be verified that v_k (in Algorithm 1) is i_n and we have r-deg(i_n) = 1 with $\alpha_1 = 0$, $\alpha_2 = \pi$, and $\gamma = \pi/2$ (in Theorem 1). Therefore, one can write $j = g_1 R_x(-\pi/2)h$. It results in $g_1 = i_1 i_2 \cdots i_{n-1} \oplus i_n$ and $h = [i_1 i_2 \cdots i_{n-1}] R_x(\pi/2)[i_n R_x(\pi/2)j]$. Now, g_1 is an n-qubit Toffoli gate and can be decomposed independently following the same approach. To decompose h, one can verify that $v_k = i_{n-1}$ in Algorithm 1 with r-deg(i_{n-1}) = 1, $\alpha_1 = 0$, $\alpha_2 = \pi/2$, and $\gamma = \pi/4$. Accordingly, we can write $h = g_2 R_x(-\pi/4)h_1$. Applying Algorithm 2 reveals that g_2 is an $(n-1)$-qubit Toffoli gate with i_{n-1} as the target and $i_1, i_2, \cdots, i_{n-2}$ as controls. By using the `apply` operator, $h_1 = g_2 R_x(\pi/4)h$ which leaves $v_k = i_{n-2}$. Altogether, we can write:

[7]One may set $\alpha_1 = \pi$ and $\alpha_2 = 0$. This combination generates a different circuit with the same functionality.
[8]RbDD of g_1 can be obtained by using Algorithm 2 and RbDD of s — no need to construct RbDD of g. However, an interested reader can verify that an indirect approach to construct RbDD of g (and hence the function of g) is to replace v_k by $\hat{0}$ in RbDD of g_1 which is constructed from applying Algorithm 2.

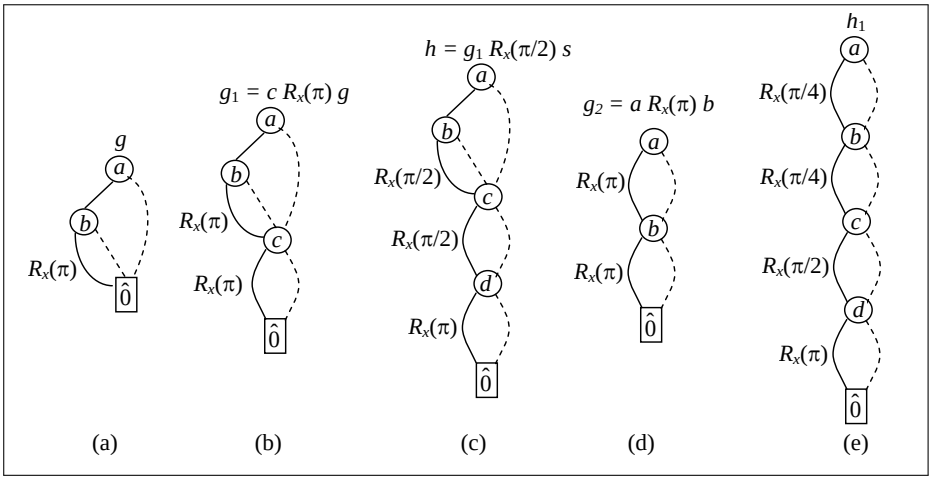

Fig. 9. RbDDs required to synthesize the 4-input Toffoli gate in Fig. 8.

$$
\begin{aligned}
C^n R_x(\pi) = C^{n-1}R_x(\pi)R_x(-\pi/2)&\\
&[C^{n-2}R_x(\pi)R_x(-\pi/4)\\
&\quad[C^{n-3}R_x(\pi)R_x(-\pi/8)\\
&\qquad[\cdots\\
&\qquad\quad[C^{n-(n-1)}R_x(\pi)R_x(-\pi/2^{n-1})\\
&[i_1R_x(\pi/2^{n-1})(i_2R_x(\pi/2^{n-1})(i_3R_x(\pi/2^{n-2})(\cdots(i_nR_x(\pi/2)j)\cdots)))]\\
&]\cdots]]]
\end{aligned}
$$

To construct the circuit, for $[i_1R_x(\pi/2^{n-1})(i_2R_x(\pi/2^{n-1})(\cdots(i_nR_x(\pi/2)j)\cdots))]$ one needs to add n controlled-rotation gates with controls on i_1, i_2, $\cdots$, i_n and targets on j. This subcircuit should be followed by constructing a $g_1 = C^{n-1}R_x(\pi)$ gate which automatically constructs all $g_2 = C^{n-2}R_x(\pi)$, $g_3 = C^{n-3}R_x(\pi)$, $\cdots$, $g_n = CR_x(\pi)$ gates too. Next, one needs to use $n-1$ controlled-rotation gates with controls on g_2, g_3, $\cdots$, g_n and targets on j. Altogether, we need $\mathrm{COST}_{1,\mathrm{C^nNOT}} = 2\mathrm{n} - 1 + \mathrm{COST}_{1,\mathrm{C^{n-1}NOT}}$ controlled-rotation gates to implement a $\mathrm{C}^n\mathrm{NOT}$ gate. To restore i_1, i_2, $\cdots$, i_n qubits to their original values, additional cost should be applied which is $\mathrm{COST}_{2,\mathrm{C^nNOT}} = \mathrm{COST}_{1,\mathrm{C^{n-1}NOT}}$, i.e., all gates excluding gates with targets on j. Terminal conditions are $\mathrm{COST}_{1,\mathrm{C^2NOT}} = 4$ and $\mathrm{COST}_{2,\mathrm{C^2NOT}} = 1$ (see Fig. 2(c)). Total implementation cost is $\mathrm{COST}_{\mathrm{C^nNOT}} = \mathrm{COST}_{1,\mathrm{C^nNOT}} + \mathrm{COST}_{2,\mathrm{C^nNOT}}$ which is *polynomial*, i.e., $2n^2-2n+1$. Fig. 11 illustrates this construction for a 5-input Toffoli gate. No ancilla is required in the proposed construction. Current constructions for a C^nNOT gate use an *exponential* number of 2-qubit gates $2^{n+1} - 3$ [13, Lemma 7.1] or $48n^2 + O(n)$ arbitrary 2-qubit operations [13, Lemma 7.6], if no ancilla is available. ∎

Quantum adder. Consider a **full adder** with inputs x_1, x_2, and x_3 ($x_1 < x_2 < x_3$) and outputs $s = x_1 \oplus x_2 \oplus x_3$ and $c = x_1x_2 + x_1x_3 + x_2x_3$. The RbDDs of s and c are shown in Fig.

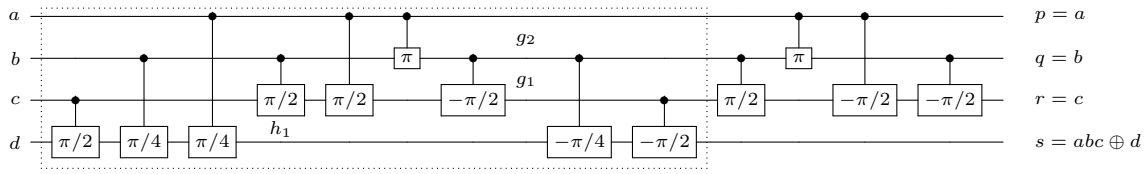

Fig. 10. Automatic synthesis of a 4-input Toffoli gate obtained by the `factor` algorithm. Only rotation angles are reported for $R_x(\theta)$ gates. The first subcircuit generates output s in Fig. 8 whereas the remaining gates generate outputs p, q and r.

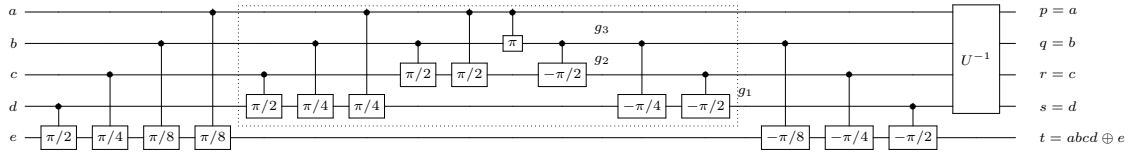

Fig. 11. Synthesized circuit for a 5-input Toffoli gate. Only rotation angles are reported for $R_x(\theta)$ gates. The dashed subcircuit generates output for a 4-input Toffoli gate (see Fig. 10) and U^{-1} is the reverse of this subcircuit.

12(a). The RbDD of s has a chain structure that corresponds to a cascade expression and can be directly realized. On the other hand, the RbDD of c should be recursively decomposed by using Algorithm 1. Using this algorithm, c is bi-decomposed as $c = g_1 R_x(-\pi/2)h$.

To construct RbDD of g_1 note that $v_k = x_3$. Applying Algorithm 2 leads to four internal nodes as follows. Node n_1 with the decision variable x_1, $w_1 = w_0 = I$, and $\hat{1}$-child node n_2, and $\hat{0}$-child node n_3. Node n_2 with the decision variable x_2, node weight $R_x(\pi)$, $w_1 = R_x(\pi)$, $w_0 = I$, and node n_4 as both $\hat{1}$-child and $\hat{0}$-child. Node n_3 with the decision variable x_2, $w_1 = R_x(\pi)$, $w_0 = I$, and node n_4 as both $\hat{1}$-child and $\hat{0}$-child. Node n_4 with decision variable x_3 connected to the terminal node $\hat{0}$ with $w_1 = R_x(\pi)$, and $w_0 = I$. A careful consideration reveals that this RbDD can be converted to the one constructed for s in Fig. 12(a). Therefore, g_1 has a cascade expression and a realizable rotation-based implementation. Finally, the RbDD for $h = g_1 R_x(\pi/2)c$ is shown in Fig. 12(a). As can be seen, the RbDD of h has a chain structure too. The resulting quantum circuit is depicted in Fig. 12(b).

Now consider a **2-qubit quantum adder** with inputs a_1, a_0, b_1, b_0 for $a_0 < b_0 < a_1 < b_1$ and outputs c, s_1, and s_0 for $s_0 < s_1 < c$. It can be verified that $s_0 = b_0 \oplus a_0, s_1 = a_0 b_0 \oplus a_1 \oplus b_1, c = a_0 b_0 a_1 \oplus a_0 b_0 b_1 \oplus a_1 b_1$. Applying the above approach leads to the following equations:

$$
\begin{aligned}
&s_0 = a_0 R_x(\pi) b_0 \\
&\begin{cases} s_1 = g_1 R_x(-\pi/2) h_1 \\ g_1 = s_0 \\ h_1 = a_0 R_x(\pi/2)(b_0 R_x(\pi/2)(a_1 R_x(\pi) b_1)) \end{cases} \\
&\begin{cases} c = g_2 R_x(-\pi/2) h_2 \\ g_2 = s_1 \\ h_2 = g_3 R_x(-\pi/2) h_3 \\ g_3 = s_0 \\ h_3 = a_0 R_x(\pi/4)(b_0 R_x(\pi/4)(a_1 R_x(\pi/2)(b_1 R_x(\pi)\hat{0}))) \end{cases}
\end{aligned}
$$

Therefore, s_0, s_1, and c can be implemented by one, four, and six 2-qubit gates (11 in total), respectively. The circuit uses one ancilla for c; a_0, a_1 remain unchanged and s_1 and s_0 are constructed on b_1 and b_0, respectively.

To generalize, consider an n-**qubit quantum ripple adder** with inputs a_i and b_i and outputs s_i and c for $0 \le i \le n-1$ and $a_0 < b_0 < a_1 < b_1 < \cdots < a_{n-1} < b_{n-1}$ and $s_0 < s_1 < \cdots < s_{n-1} < c$. We have:

$$
\begin{aligned}
&s_0 = a_0 R_x(\pi) b_0 \\
&s_1 = s_0 R_x(-\pi/2)(a_0 R_x(\pi/2)(b_0 R_x(\pi/2)(a_1 R_x(\pi)(b_1 R_x(\pi)\hat{0})))) \\
&\cdots
\end{aligned}
$$

$$
\begin{aligned}
s_{n-1} = s_{n-2} R_x(-\pi/2)(s_{n-3} R_x(-\pi/2)(\cdots(s_0 R_x(-\pi/2) \\
&(a_0 R_x(\pi/2^{n-1})(b_0 R_x(\pi/2^{n-1}) \\
&(a_1 R_x(\pi/2^{n-2})(b_1 R_x(\pi/2^{n-2}) \\
&\cdots \\
&(a_{n-2} R_x(\pi/2)(b_{n-2} R_x(\pi/2)) \\
&(a_{n-1} R_x(\pi)(b_{n-1} R_x(\pi)\hat{0})))\cdots) \\
c = s_{n-1} R_x(-\pi/2)(s_{n-2} R_x(-\pi/2)(\cdots(s_0 R_x(-\pi/2) \\
&(a_0 R_x(\pi/2^{n})(b_0 R_x(\pi/2^{n}) \\
&(a_1 R_x(\pi/2^{n-1})(b_1 R_x(\pi/2^{n-1}) \\
&\cdots \\
&(a_{n-2} R_x(\pi/4)(b_{n-2} R_x(\pi/4)) \\
&(a_{n-1} R_x(\pi/2)(b_{n-1} R_x(\pi)\hat{0})))\cdots)
\end{aligned}
$$

To count the number of 2-qubit gates, note that there are $2n$ gates on c, $2n-1$ gates on b_{n-1}, $2n-3$ gates on b_{n-2}, $\cdots$, 3 gates on b_1 and 1 gate on b_0 in the proposed construction. This subcircuit should be followed by a 2-qubit gate conditioned on b_0 with target on b_1, 2 gates conditioned on b_0 and b_1 with targets on b_2, 3 gates conditioned on b_0, b_1, b_2 with targets on b_4, etc. Altogether, an n-qubit quantum ripple adder can be implemented with $1/2(3n^2+5n)$ controlled-rotation gates. Fig. 13 illustrates the proposed construction for a 5-qubit carry-ripple adder. This circuit is restructured in Fig. 14 with parallel gates. Compared to the construction in [14, Figure 7] with depth 28, our circuit uses a wider varieties of rotation

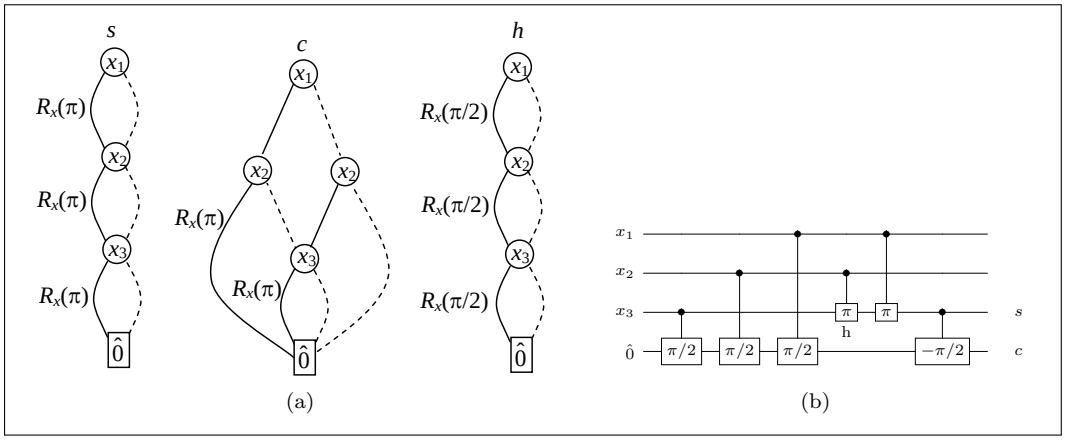

Fig. 12. (a) RbDDs for a 3-qubit full adder and (b) the synthesized circuit. Only rotation angles are reported for $R_x(\theta)$ gates.

angles to reduce the depth to 23. Circuit depth for $n = 2, \cdots, 15$ is 9(10), 12(16), 19(22), 23(28), 27(34), 31(40), 39(46), 43(52), 48(58), 51(64), 57(70), 61(76), 66(82), 70(88) where $a(b)$ denotes a 2-qubit gates in the proposed construction and b 2-qubit gates in [14].[9] ∎

Quantum multiplexer. Consider a **3-input multiplexer** $f = sx_1 + \bar{s}x_2$ with $s < x_1 < x_2$. Following Theorem 1 leads to $\alpha_1 = 0, \alpha_2 = \pi, \gamma = \pi/2$ and $f = g_1 R_x(\pi/2)h$ for $v_k = x_2$ with r-deg$(x_2) = 1$. To construct g_1 note that we use $\alpha_1 = 0$. It results in a chain structure for g_1 with the factored form $g_1 = sR_x(\pi)x_2$.

To construct the RbDD of $h = g_1 R_x(-\pi/2)f$ using the `apply` operator, note that both g_1 and f use s. Accordingly, one needs to apply Rule 3 which results in $w_1 = [R_x(\pi)x_2]R_x(-\pi/2)x_1$ and $w_0 = x_2 R_x(-\pi/2)x_2 = x_2 R_x(\pi/2)\hat{0}$. Continuing this path results in $h = g_2 R_x(-\pi/2)h_1$, $g_2 = sR_x(\pi)x_1$, and $h_1 = x_1 R_x(\pi/2)[x_2 R_x(\pi/2)\hat{0}]$.

Altogether, $f = [sR_x(\pi)x_2]R_x(-\pi/2)[[sR_x(\pi)x_1]R_x(-\pi/2)[x_1 R_x(\pi/2)[x_2 R_x(\pi/2)\hat{0}]]]$. The resulting quantum circuit is shown in Fig. 15(a). Note that one $\hat{0}$-initialized qubit is added to setup $[x_2 R_x(\pi/2)\hat{0}]$.

The **6-input multiplexer** $f = s_1 s_2 x_1 + s_1 \bar{s}_2 x_2 + \bar{s}_1 s_2 x_3 + \bar{s}_1 \bar{s}_2 x_4$ can be constructed by using three 3-input multiplexers, which would require 3 extra ancillae. However, if the `factor` algorithm is directly applied, the resulting circuit only uses one ancilla as illustrated in Fig. 15(b). For an **n-qubit quantum multiplexer** with $\lceil \log n \rceil$ selects and n inputs, the proposed approach leads to $2n$ 2-qubit gates and n C$^{\lceil \log n \rceil} R_x(\pi)$ gates with one ancilla. As the cost of an n-qubit multiple-control Toffoli gate is $2n^2 - 2n + 1$, the proposed approach leads to $O(n \log^2 n)$ gates, i.e., $2n + n(2\lceil \log n \rceil^2 - 2\lceil \log n \rceil + 1)$. We found no explicit construction for an n-qubit quantum multiplexer in the literature. However, one can use n $C^{\lceil \log n \rceil + 1}$CNOT gates in a circuit with one zero-initialized ancilla to implement an n-qubit multiplexer. Considering linear-size cost for each gate [13] leads to $O(n^2)$ cost.[10] ∎

[9]The construction in [14] generates a circuit with controlled-rotation gates with phase $\pi/2$ and total depth $6n - 2$ for an n-qubit carry-ripple adder. We guess our circuit depth is $5n + O(1)$. The trend line for the number of 2-qubit gates in the proposed construction for $n = 2, \cdots, 15$ is $4.7868n - 0.9736$.

[10]For example, a 4-to-1 multiplexer can be implemented as $T(s_0', s_1', x_0, f)$, $T(s_0', s_1, x_1, f)$, $T(s_0, s_1', x_2, f)$,

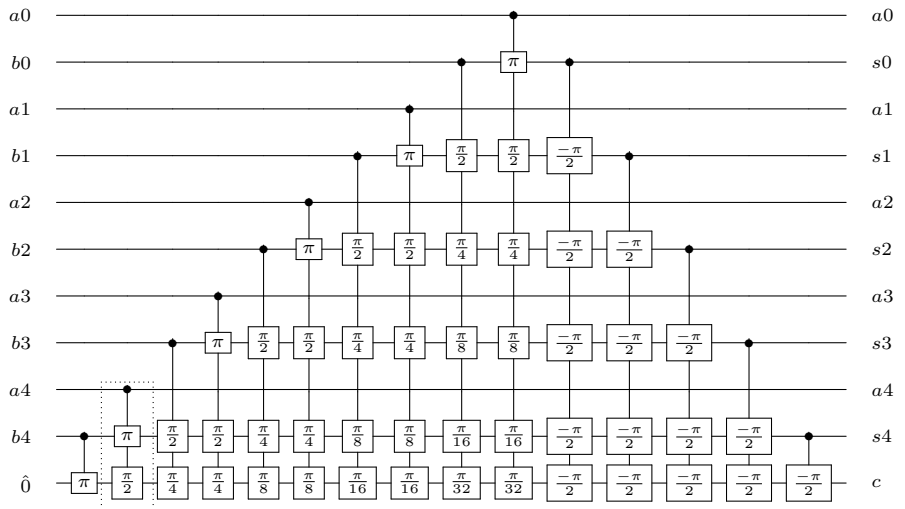

Fig. 13. 5-bit ripple-carry adder synthesized by the proposed approach with 50 2-qubit rotation gates. Different gates with the same control line are represented as one gate with several targets. For example, the gate in the dashed box includes two rotation gates conditioned on a_4 with targets on c (ancilla) and b_4.

Quantum Fourier Transform. The **quantum Fourier transform (QFT)** is used in many quantum algorithms. QFT has an efficient quantum circuit implementation based on R_z gates [1]. The result of applying QFT on inputs $j_1, j_2, \ldots, j_n$ is $|0\rangle + e^{2\pi i 0.j_n}|1\rangle$, $|0\rangle + e^{2\pi i 0.j_2 \cdots j_n}|1\rangle$, $\cdots$, $|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle$ where the common notation $0.j_1 j_2 \ldots j_n = \frac{j_1}{2} + \frac{j_2}{2^2} + \cdots + \frac{j_n}{2^n}$ is used.

Following the discussion in Section 5, in the first step the output $f_n(j_1, j_2, \ldots, j_n) = |0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle$ is described by $f_n(J) = e^{i\delta(J)} R_z(\gamma(J)) R_x(\theta(J))|0\rangle$ where $J = j_1 2^{n-1} + j_2 2^{n-2} + \cdots + j_n$. For this function $\theta(J) = (\pi/2) j_1$, and $g(J) = R_x(\theta(J))\hat{0} = R_x(\pi/2)\hat{0}$.

The RbDD for $R_z(\gamma(J))$ is shown in Fig. 16(a) where the root node is weighted. This RbDD corresponds to a cascade expression and there is no need to perform bi-decomposition. The quantum circuit implementing $f_n(J) = e^{i\delta(J)} R_z(\gamma(J)) R_x(\theta(J))|0\rangle$ in shown in Fig. 16(b).

The single qubit operation $R_z(-\pi/2)$ can be moved between $R_x(\pi/2)$ and controlled $R_z(\pi)$ operations. Since j_1 is used as the controlled qubit of only one controlled rotation operation, the sub-circuit in Fig. 17(a) can be replaced by a single qubit operator shown in Fig. 17(b). The 2×2 matrix describing U consists of two columns u_0 and u_1 such that $U = [\ u_0 \quad u_1\ ]$ and can be obtained as follows:

$$u_0 = R_z(-\pi/2) R_x(\pi/2)|0\rangle = \frac{e^{-i\pi/4}}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$u_1 = R_z(\pi) R_z(-\pi/2) R_x(\pi/2)|0\rangle = R_z(\pi/2) R_x(\pi/2)|0\rangle = \frac{e^{-i\pi/4}}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$T(s_0, s_1', x_3, f)$ for a circuit with selects s_0, s_1, inputs x_0, x_1, x_2, x_3 and output f, where f is a zero-initialized ancilla. For each T (Toffoli) gate, the first three lines act as the control lines and the last line acts as the target. In addition, e.g., $T(s_0', s_1', x_0, f)$ applies x_0 on f when $s_0 = 0, s_1 = 0$. This can be implemented by N(s_0), N(s_1), $T(s_0, s_1, x_0, f)$, N(s_1), $N(s_0)$ where N denotes the NOT gate.

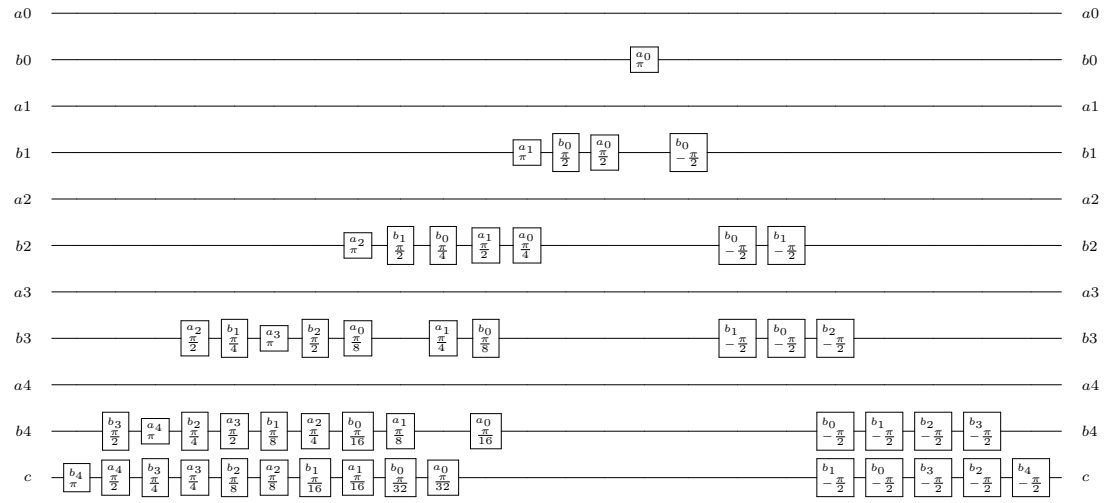

Fig. 14. Circuit in Fig. 13 with depth 23. The notation $\boxed{\begin{smallmatrix}a\\ \theta\end{smallmatrix}}$ represents controlled-rotation gates where a is the control line and θ is the rotation angle.

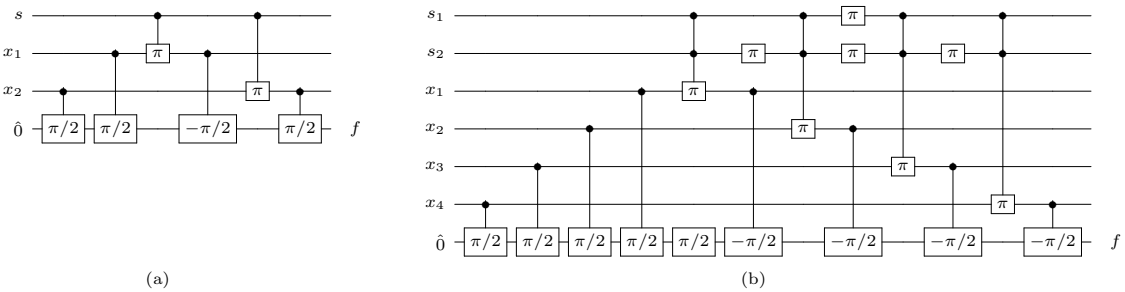

Fig. 15. (a) Quantum 2-to-1 multiplexer, (b) quantum 4-to-1 multiplexer. Circuits are directly obtained by the `factor` algorithm. Only rotation angles are reported for $R_x(\theta)$ gates.

Hence, we have:

$$U = \frac{e^{-i\pi/4}}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

This operator can be replaced by the Hadamard operator since the two operators differ only in a global phase. Therefore, the quantum circuit for $|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle$ can be realized as shown in Fig. 17(c). The remaining outputs can be generated similarly. Accordingly, the proposed method results in the same circuit structure in [1] with $n(n+1)/2$ total gates. This can show the efficiency of the proposed *automatic* synthesis approach. ∎

7 Conclusions and Further Discussion

We mainly addressed reversible logic synthesis by quantum rotation-based gates. A new canonical representation model was proposed based on binary decision diagrams. Focused on it, we developed a synthesis framework to manipulate circuits and to synthesize functions with binary variables. We also showed that the proposed approach can be extended to work with functions that generate arbitrary outputs for binary inputs.

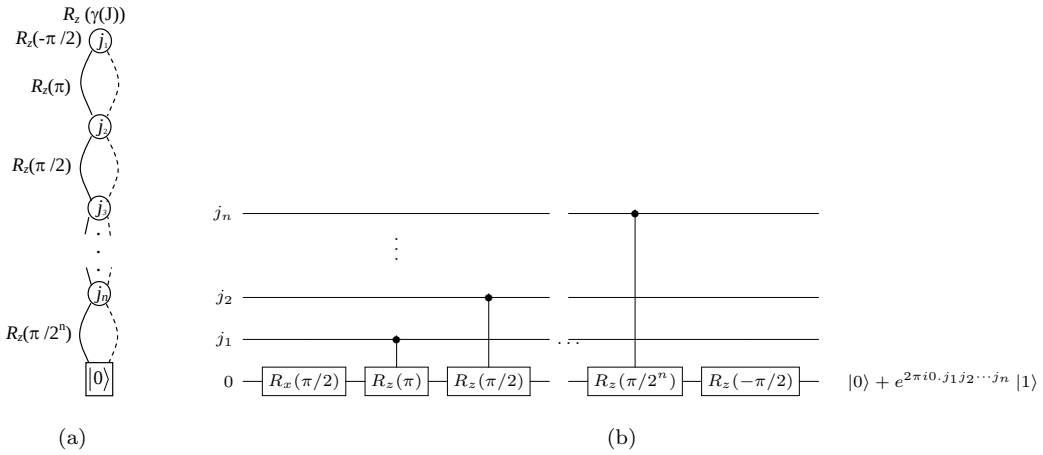

Fig. 16. (a) The RbDD for $R_z(\gamma(J))$ in QFT. (b) Quantum circuit for $|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle$ in the quantum Fourier transform.

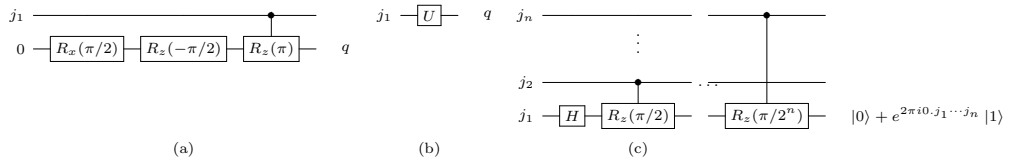

Fig. 17. Replacing a part of the QFT circuit (a) in Fig. 16(b) with a single qubit operator shown in (b) leads to the QFT circuit given in (c) for one output of QFT.

While almost all previous synthesis methods with favorable results [7] used CNOT, controlled-V and controlled-V† gates (see Fig. 1) as primitive gates with unit cost, we used 2-qubit controlled-rotation gates. This work can be particularly considered as a synthesis method for Boolean reversible circuits that computes a given Boolean function outside the Boolean domain with quantum gates [12]. We hope this new insight opens further analysis and investigation to efficiently address quantum and reversible logic synthesis possibly beyond current achievements [7].

To realize a given quantum computation by fault-tolerant gates, one needs to use those gates that have direct fault-tolerant implementations [1]. Such realizations are only available for a few operations such as Clifford gates. To implement a wider set of gates such as the ones we used in this paper, one must apply the set of fault-tolerant gates to accurately (by approximation) implement other gates. This can be done by the Solovay-Kitaev algorithm [1]. Given the point that the proposed approach uses controlled rotation gates with various angles, fault-tolerant implementation of the proposed circuits can be costly. Future work should address this issue. Additionally, further progress on this path may result in new observations to restrict/ignore angles [32, 33] and to remove redundant gates.

Acknowledgements

MS thanks Alireza Shafaei for useful discussion. Authors were supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National

Business Center contract number D11PC20165. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

1. M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.
2. V. Giovannetti, S. Lloyd, and L. Maccone. Advances in quantum metrology. *Nature Photonics*, 5(4):222–229, 2011.
3. C. Kothe, G. Björk, S. Inoue, and M. Bourennane. On the efficiency of quantum lithography. *New Journal of Physics*, 13(4):043028, 2011.
4. B. P. Lanyon et al. Universal digital quantum simulation with trapped ions. *Science*, 334(6052):57–61, Oct 2011.
5. S. Aaronson and A. Ambainis. The need for structure in quantum speedups. *Symp. on Innovations in Comput. Science*, pages 338–352, 2011.
6. K. R. Brown et al. Single-qubit-gate error below 10^{-4} in a trapped ion. *Phys. Rev. A*, 84:030303, Sep 2011.
7. M. Saeedi and I. L. Markov. Synthesis and optimization of reversible circuits - a survey. *ACM Computing Surveys, arXiv:1110.2574*, 2013.
8. V.V. Shende, S.S. Bullock, and I. L. Markov. Synthesis of quantum-logic circuits. *IEEE Trans. CAD*, 25(6):1000–1010, Jun 2006.
9. I. L. Markov and M. Saeedi. Constant-optimized quantum circuits for modular multiplication and exponentiation. *Quantum Info. Comput.*, 12(5-6):361–394, May 2012.
10. I. L. Markov and M. Saeedi. Faster quantum number factoring via circuit synthesis. *Phys. Rev. A*, 87:012310, Jan 2013.
11. H. J. Garcia, I. L. Markov, and A. W. Cross. Efficient inner-product algorithm for stabilizer states. *arXiv:1210.6646*, 2012.
12. D. Maslov and M. Saeedi. Reversible circuit optimization via leaving the Boolean domain. *IEEE Trans. CAD*, 30(6):806 – 816, Jun. 2011.
13. A. Barenco et al. Elementary gates for quantum computation. *Physical Review A*, 52:3457–3467, 1995.
14. S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton. A new quantum ripple-carry addition circuit. *arXiv:0410184*, 2004.
15. A. Abdollahi and M. Pedram. Analysis and synthesis of quantum circuits by using quantum decision diagrams. In *Design, Automation and Test in Europe*, pages 317–322, Mar. 2006.
16. D. Maslov, G. W. Dueck, and D. M. Miller. Techniques for the synthesis of reversible Toffoli networks. *ACM Trans. Des. Autom. Electron. Sys.*, 12(4):42:1–42:28, Sep 2007.
17. J. Donald and N. K. Jha. Reversible logic synthesis with Fredkin and Peres gates. *J. Emerg. Technol. Comput. Sys.*, 4(1):2:1–2:19, Apr 2008.
18. M. Saeedi, M. Saheb Zamani, M. Sedighi, and Z. Sasanian. Reversible circuit synthesis using a cycle-based approach. *J. Emerg. Technol. Comput. Sys.*, 6(4):13:1–13:26, Dec. 2010.
19. R. Wille and R. Drechsler. BDD-based synthesis of reversible logic for large functions. In *Design Automation Conference*, pages 270–275, 2009.
20. M. Möttönen and J. J. Vartiainen. *Decompositions of general quantum gates*. Ch. 7 in Trends in Quantum Computing Research, NOVA Publishers, 2006.
21. V. Bergholm, J. J. Vartiainen, M. Möttönen, and M. M. Salomaa. Quantum circuits with uniformly controlled one-qubit gates. *Phys. Rev. A*, 71:052330, 2005.
22. V. V. Shende, I. L. Markov, and S. S. Bullock. Minimal universal two-qubit quantum circuits.

Phys. Rev. A, 69:062321, 2004.
23. M. Saeedi, M. Arabzadeh, M. Saheb Zamani, and M. Sedighi. Block-based quantum-logic synthesis. *Quant. Inf. Comput.*, 11(3-4):0262–0277, 2011.
24. C. Y. Lee. Representation of switching circuits by binary decision programs. *Bell System Technical Journal*, 38(4):985–999, 1959.
25. S. B. Akers. Functional testing with binary decision diagrams. *Annual Conference of Fault-Tolerant Computing*, pages 75–82, 1978.
26. R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35(8):677–691, August 1986.
27. G. F. Viamontes, I. L. Markov, and J. P. Hayes. Checking equivalence of quantum circuits and states. In *IEEE/ACM international conference on Computer-aided design*, pages 69–74, 2007.
28. S. Yamashita, S. Minato, and D. M. Miller. DDMF: An efficient decision diagram structure for design verification of quantum circuits under a practical restriction. *IEICE Transactions*, 91-A(12):3793–3802, 2008.
29. S.-A. Wang, C.-Y. Lu, I-M. Tsai, and S.-Y. Kuo. An XQDD-based verification method for quantum circuits. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, E91-A(2):584–594, February 2008.
30. D. M. Miller and M. A. Thornton. QMDD: A decision diagram structure for reversible and quantum circuits. *Int'l Symp. on Multiple-Valued Logic*, page 30, 2006.
31. G. F. Viamontes, I. L. Markov, and J. P. Hayes. *Quantum Circuit Simulation*. Springer, 2009.
32. A. Barenco, A. Ekert, K.-A. Suominen, and P. Törmä. Approximate quantum Fourier transform and decoherence. *Phys. Rev. A*, 54:139–146, Jul 1996.
33. A. G. Fowler and L. C. L. Hollenberg. Scalability of Shor's algorithm with a limited set of rotation gates. *Phys. Rev. A*, 70:032329, Sep 2004.