



An Efficient Design of Reversible Multi-Bit Quantum Comparator Via Only a Single Ancillary Bit

Haiying Xia^{1,2,3} · Haisheng Li¹ · Han Zhang¹ · Yan Liang¹ · Jing Xin⁴

Received: 16 April 2018 / Accepted: 7 September 2018 / Published online: 14 September 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract

Quantum computing has emerged as one of the most promising technology due to its powerful computing capability. And quantum basic circuits like quantum comparator, quantum adder etc, are the foundation to realize quantum computing. In this paper, we present an efficient design to realize the comparison of two n -bit quantum logic states via only a single ancillary bit. Our proposed comparator compares two n -bit quantum logic states and identifies which of them is the largest, which of them is the smallest, and which of them is equal in linear quantum depth. Moreover, we analyze the superior performance of our proposed comparator in terms of auxiliary bits compared with the existing quantum logic comparators.

Keywords n -bit quantum logic comparator · Ancillary bit · Efficient

✉ Haisheng Li
187373363@qq.com

Haiying Xia
xhy22@mailbox.gxnu.edu.cn

Han Zhang
1942641420@qq.com

Yan Liang
69851012@qq.com

Jing Xin
xinj@xaut.edu.cn

¹ Guangxi Normal University, Guangxi, China

² Guilin University of Electronic Technology, Guilin, China

³ Guilin University of Aerospace Technology, Guilin, China

⁴ Xi'an University of technology, Xi'an, China

1 Introduction

Reversible computing is those which can preserve invertibility during data computing. the von Neumann-Landauer (VNL) principle tells us that the reversible logic operations (which have one-to-one mapping between pre logic operations and post logic operations) don't incur information lost [1]. Quantum computing is exactly based on this reversible theory, realized by reversible gates where each reversible gate performs a specific unitary operation [2]. Since there is no information lost, quantum computing has become a focus in the research fields of information technology [3]. Hence, the circuit design of basic quantum operation is an urgent issue to improve the development of quantum computing.

Comparator is one of most used devices for basic operations in traditional Turing machine. So it is easy to think of how to realize quantum comparator. Quantum comparator can compare two quantum logic states and identify whether they are equal or, otherwise, which of them is the largest [4]. Quantum comparator has three major design parameters: ancillary bits, quantum cost and quantum delay [5]. Ancillary bits assist the quantum circuit to achieve some specific operation. Quantum cost means the complexity of quantum comparator circuit while quantum delay is the time delay to run out the whole quantum circuit. Obviously, an optimized design of quantum comparator is the one who can compare two quantum logic states with less quantum cost and less ancillary bits in less time.

Several works have been proposed to efficiently design quantum comparators, including the serial based design [3, 4] and the tree based design [6, 7]. A serial based quantum comparator serially performs the quantum bit comparison from low bit to high bit, while a tree based quantum comparator can accomplish the comparison of quantum bits at once. Although the tree based quantum comparator has advantages over the serial based comparator in terms of time delay, the quantum cost of the tree based comparator is inferior to the serial based. As for the ancillary inputs, both of two designs involve one or two bits comparison with two or four ancillary bits. So far as we know, there are no three or more bits quantum comparator with less ancillary bits reported in literatures.

Inspired by the ripple-carry design in addition, we present an efficient design of reversible multi-bit quantum comparator. Our proposed comparator allows one comparator to perform one or more quantum bit comparison simultaneously with only a single ancillary bit. Meanwhile, our proposed comparator guarantees the quantum bits unchanged. In a sum, compared with existing quantum comparators, our proposed comparator has three contributions. (1) Our proposed comparator involves one or more quantum logic bits comparison with only a single ancillary bit. (2) Our comparator has relatively less quantum cost compared with existing quantum logic comparators with unchanged quantum bits. (3) The performance of our proposed comparator is completely analyzed in terms of quantum cost, quantum delay and ancillary bits.

The rest of this paper is organized as follows. In Section 2, we introduce the general concepts of quantum computation, especially the existing quantum comparators. In Section 3, our multi-bit comparator is proposed and analyzed in detail. Section 4 presents the comparison and discussion of the existing comparators with our proposed comparator in terms of quantum cost, quantum delay and ancillary bits. Conclusions are drawn in Section 5.

2 Related Works

2.1 Quantum Gates

Although there are many quantum gates proposed in literatures, these quantum gates are often designed based on several basic quantum gates such as NOT gate, Controlled NOT (CNOT) gate, swap gate and V gate. Since each of these quantum gates performs a specific unitary operation, each quantum gate can be represented by a unitary matrix [8, 9]. If the quantum gate acts on i quantum bits where a quantum bit, works on quantum circuits instead of classical circuits as a bit performed on traditional Turing machine, then, this quantum gate is represented by the unitary matrix [10, 11]. Generally, NOT gate and Controlled NOT gate is most basic gate with quantum cost of 1Δ and quantum delay of 1Δ . Figure 1 lists four basic quantum gates.

In Fig. 1, the black dot in CNOT gate denotes when the controlled bit (black dot) is “1”, the NOT gate in the other bit works, while the white dot denotes when the controlled bit is “0”, the NOT gate works. The controlled bits i_1, i_2 in Toffoli gate is a binary string, including four cases: “00”, “01”, “10”, and “11”.

V and $V+$ are another two popular gates with quantum cost of 1Δ and quantum delay of 1Δ [12]. They are represented by two unitary matrices, satisfied with $VV+$ or $V+V=1$ or VV or $V+V+=\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$



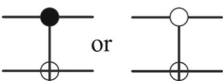
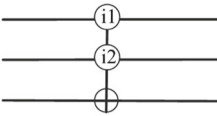

Name	Quantum circuit representation
One qubit	
NOT gate	
CNOT gate	
Toffoli gate	
Swap gate	

Fig. 1 Some basic quantum gates

The above V and $V+$ gates can be controlled by another input bit, called controlled V gate or controlled $V+$ gate. Hence, Toffoli gate and Swap gate can be designed by controlled V gate, controlled $V+$ gate, and CNOT gate. Figure 2a is the design of Toffoli gate and Fig. 2b is Swap gate. From Fig. 2, the quantum cost of Toffoli gate is decided by the number of CNOT gate, controlled V or controlled $V+$ gate. There are two CNOT gates, two controlled V gates and one controlled $V+$ gate in Toffoli gate. So the quantum cost and delay of Toffoli gate are both 5Δ . Obviously, the quantum cost and delay of Swap gate are both 3Δ .

2.2 Related Works on Quantum Comparators

There are several quantum comparators reported. Wang et al. presented a quantum comparator which realized the comparison of two quantum logic states with n quantum bits in a serial way [3]. The circuit has $2n - 2$ ancillary input bits as shown in Fig. 3, where $|e_0\rangle$ and $|e_1\rangle$ are useful outputs. Al-Rabadi proposed a serial based quantum comparator which cascaded a chain of 1-bit comparator given in Fig. 4, which has 6 ancillary input bits for 1-bit comparator [4]. Thapliyal et al. designed a tree-based comparator shown in Fig. 5a in which each node is a 2-bit comparator shown in Fig. 5b that can compare 2-bit binary numbers [5]. Vudadha et al. further improved the tree-based comparator based on a prefix tree [6]. This design has three stages. The first stage consists of a 1-bit comparator with two useful outputs. The outputs of 1-bit comparator stage are grouped in the second stage using prefix grouping and the final outputs G are generated. Although the tree-based quantum comparator is superior to the serial-based comparator in time delay, it is inferior to the serial-based in ancillary bits.

2.3 Related Works on Quantum Subtractors

In fact, quantum subtractor can also identify whether one quantum logic state is bigger than another quantum logic state by examining the highest ancillary input bit is $|0\rangle$ or $|1\rangle$. Quantum subtractor generally uses one ancillary input to hold the borrow information for

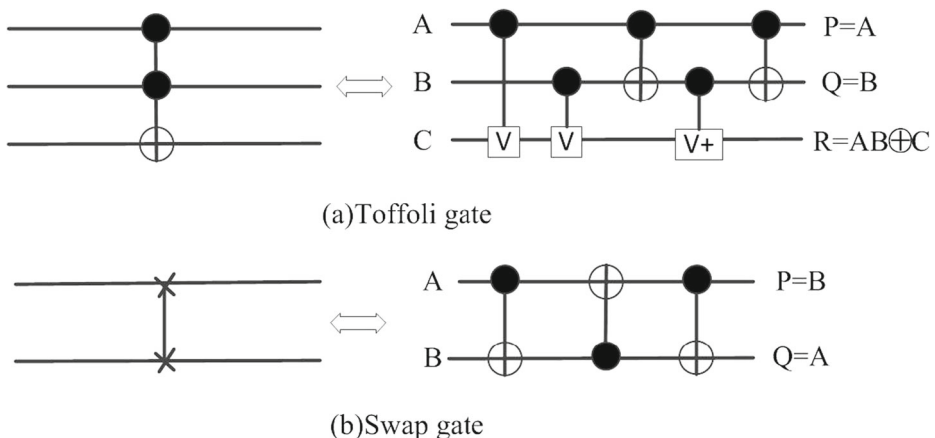


Fig. 2 The design of Toffoli gate and Swap gate

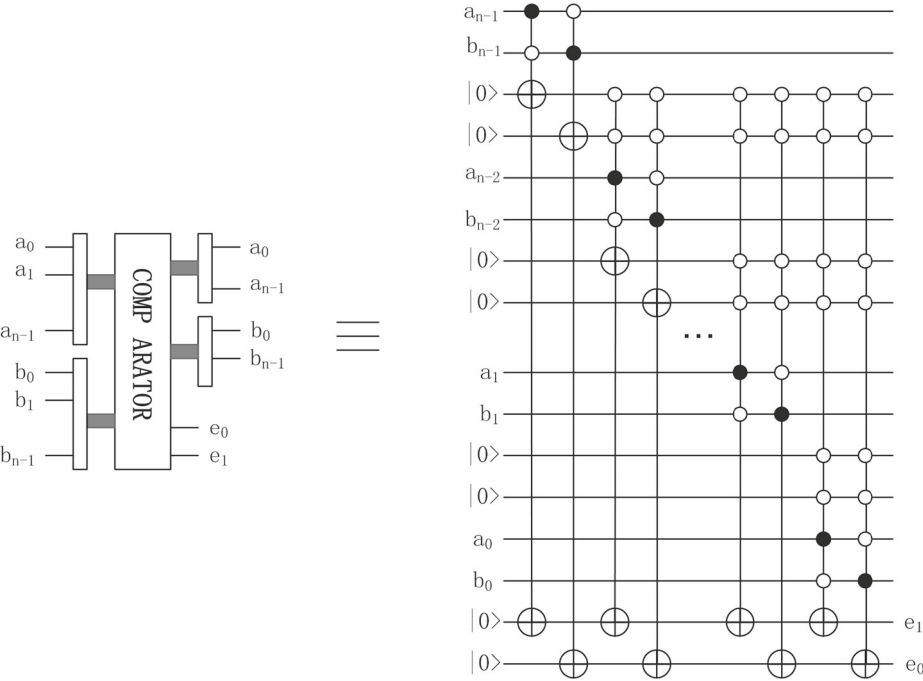


Fig. 3 The reversible comparator presented by want et al.

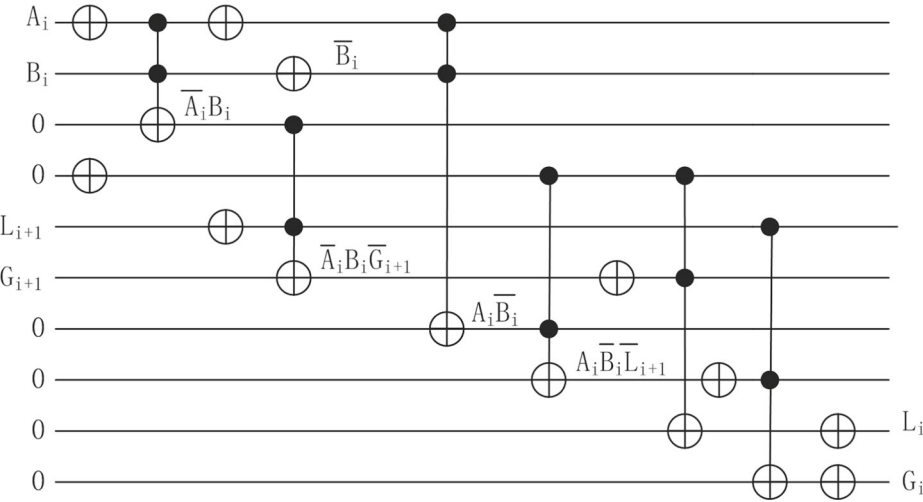


Fig. 4 The comparator presented by Al-Rabadi et al.

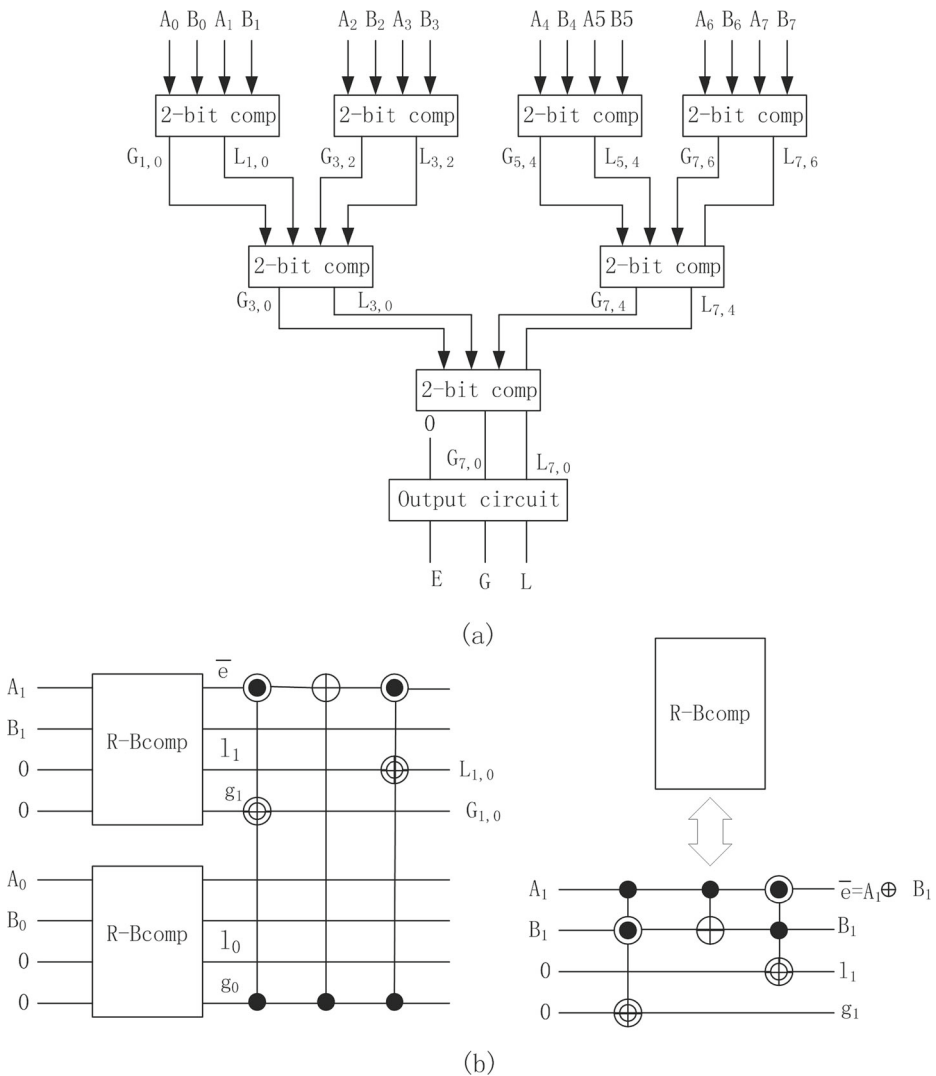


Fig. 5 The tree based comparator presented by Thapliyal et al.

1-bit subtrator. Zhou et al. constructed a 4×4 reversible gate called ZS gate to build quantum full adder with better performance compared with its counterparts [13]. Cheng et al. designed a quantum full subtrator (QFS) according to the classical truth tables [14]. Cucaro et al. presented a new linear-depth ripple-carry quantum addition circuit with only one ancillary quantum bit [15]. Draper et al. presented an efficient addition circuit, borrowing techniques from the classical carry-lookahead arithmetic circuit, which reduced the cost of addition dramatically with only slight increase in the number of required quantum bits [16]. Although some subtrators can perform the subtraction of two quantum logic states sequentially or simultaneously, they usually change one quantum logic state to save the subtraction information. So these comparison only can judge two cases, for example, $a \geq b$ or

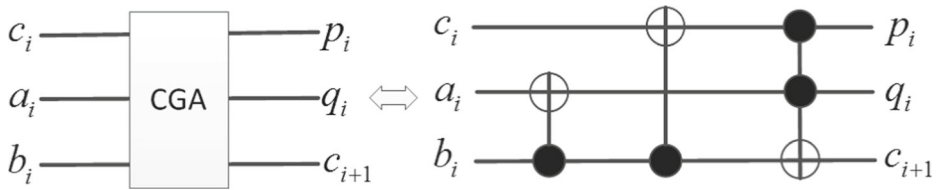


Fig. 6 The CGA gate

$a < b$. But these design technologies can give inspiration to design more efficient quantum comparators.

3 Our Proposed Multi-bit Quantum Comparator

Our aim is to compare two n -bit numbers a and b with only one auxiliary bit in average $O(N)$ operations. We use a binary string to represent $a = a_{n-1} \dots a_0$, where a_0 is the lowest-order bit. Similarly, $b = b_{n-1} \dots b_0$, where b_0 is the lowest-order bit. We use p_i , q_i and c_{i+1} to denote the outputs of c_i , a_i and b_i .

3.1 The Ripple-Carry Idea

The ripple-carry idea is successfully applied in a classical adder, where each pair of a_i and b_i is computed recursively. Assume that $c_0 = 0$, and $c_{i+1} = CGA(a_i, b_i, c_i)$ for $i \geq 0$, the c_i is computed in order from c_1 to c_n . The c_{i+1} of CG gate can carry the addition result of a_i , b_i , and c_i , satisfied with $(p_i, q_i, c_{i+1}) = CGA(a_i, b_i, c_i)$, where p_i , q_i , and c_{i+1} are represented by (1). The CGA gate is shown in Fig. 6 and it contains two CNOT gates and one Toffoli gate. So the quantum cost and delay of CGA gate are both 7Δ .

$$\begin{aligned} p_i &= b_i \oplus c_i \\ q_i &= a_i \oplus b_i \\ c_{i+1} &= (a_i \oplus b_i)(b_i \oplus c_i) \oplus b_i = a_i b_i \oplus a_i c_i \oplus b_i c_i \end{aligned} \quad (1)$$

Since the CGA gate changes the states of a_i and b_i , it needs some operations to change the states of p_i and q_i back to their initial states. Fortunately, the inverse operation of CG gate can exactly restore the states of p_i and q_i back to their initial states, named this inverse operation as ICGA (shown in Fig. 7). That means the states of a_i and b_i passing through one CGA gate and one ICGA gate remain unchanged. Assume that $B_i = ICGA(p_i, q_i, c_{i+1})$ for $i \geq 0$, the B_i is computed in order from B_n to B_0 . The ICGA gate is satisfied with

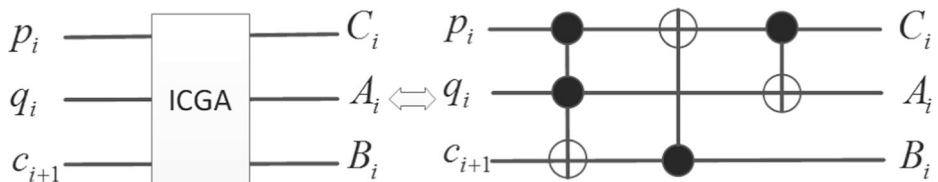


Fig. 7 The ICGA gate

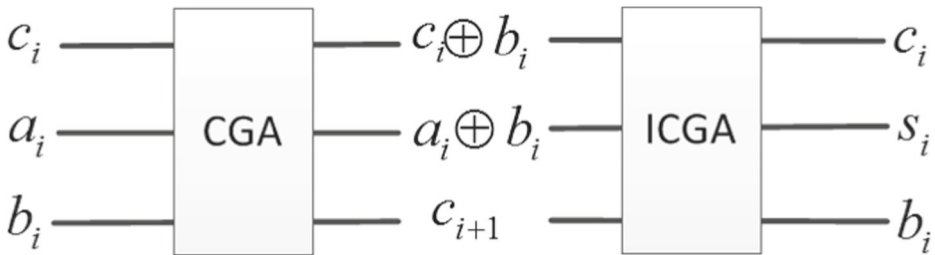


Fig. 8 The addition of c_i , a_i and b_i

$(C_i, A_i, B_i) = ICGA(p_i, q_i, c_{i+1})$, where C_i , A_i and B_i are represented by (2). The ICGA gate also has two CNOT gates and one Toffoli gate as shown in Fig. 8.

$$\begin{aligned} B_i &= p_i q_i \oplus c_{i+1} \\ C_i &= p_i \oplus p_i q_i \oplus c_{i+1} \\ A_i &= q_i \oplus p_i \oplus p_i q_i \oplus c_{i+1} \end{aligned} \quad (2)$$

Then, the addition of c_i , a_i and b_i can be realized by combining one CGA and one ICGA gate with two xor operation between CGA and ICGA gate as shown in Fig. 8. We illustrate the whole process by the case of c_i . Applying CGA gate to c_i , the carry information is written into c_{i+1} . We continue our computation. After applying the ICGA gate, the c_i is restored and written into C_i .

3.2 The Basic Ripple-Carry Gates for Our Proposed Comparator

As described in ripple-carry adder, the carry bit contains the carry information of the addition of two n -bit numbers of a and b . Similarly, the carry bit also contains the carry information of the subtraction of two n -bit numbers of a and b . If we only consider the carry information in the highest order, it indicates the comparison of a and b . For example,

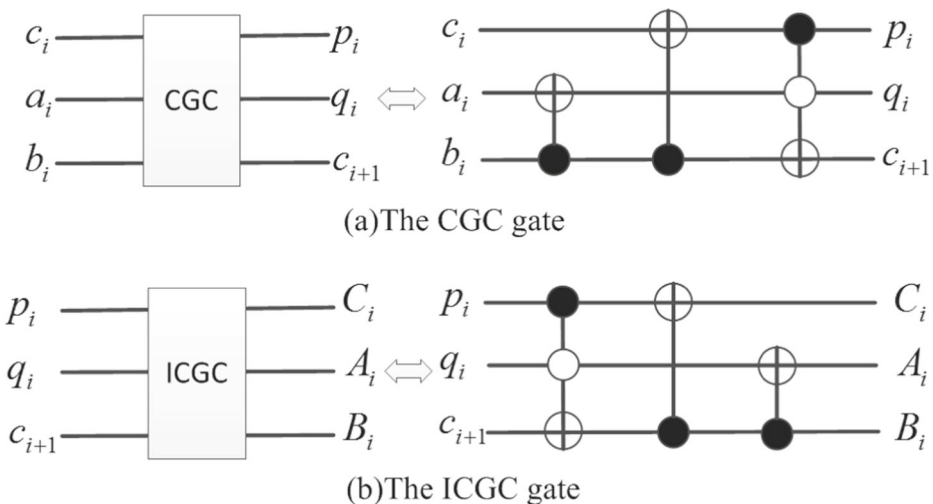


Fig. 9 The CGC and ICGC gates

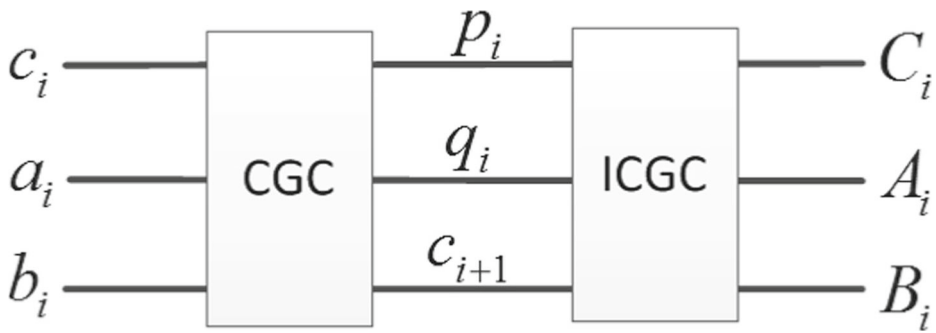


Fig. 10 The subtraction of c_i , a_i and b_i

assume that the initial state of the carry information in the highest order is 0. $a \geq b$ if the highest carry bit is 0, and vice versa.

Thus, we define the CGC gate to compute the carry information of the subtraction of a and b as shown in Fig. 9a. The CGC gate is satisfied with $(p_i, q_i, c_{i+1}) = CGC(c_i, a_i, b_i)$ for $i \geq 0$, where p_i , q_i , and c_{i+1} are represented by (3). In the same way, the ICGC gate is defined to restore the states of p_i and q_i back to their initial states as shown in Fig. 9b. The ICGC gate is satisfied with $(C_i, A_i, B_i) = ICGC(p_i, q_i, c_{i+1})$, where C_i , A_i and B_i are represented by (4). The quantum cost and quantum delay of the CGC and ICGC gate are both 7Δ .

Then, the carry information of c_i , a_i and b_i can be obtained by combining one CGC gate and one ICGC gate as shown in Fig. 10. The results of C_i , A_i , and B_i are deduced in (5) by substituting (3) into (4). After going through one CGC gate and one ICGC gate, the outputs are restored to their initials. The carry information of the subtraction of a_i and b_i are placed in c_{i+1} , and this carry information can be used to compute the subtraction in the higher order.

$$\begin{aligned} p_i &= b_i \oplus c_i \\ q_i &= b_i \oplus a_i \\ c_{i+1} &= (b_i \oplus c_i)(b_i \oplus a_i) \oplus b_i \end{aligned} \quad (3)$$

$$\begin{aligned} B_i &= p_i \bar{q}_i \oplus c_{i+1} \\ A_i &= p_i \bar{q}_i \oplus c_{i+1} \oplus q_i \\ C_i &= p_i \bar{q}_i \oplus c_{i+1} \oplus p_i \end{aligned} \quad (4)$$

$$\begin{aligned} B_i &= (b_i \oplus c_i)(b_i \oplus a_i) \oplus (b_i \oplus c_i)(b_i \oplus a_i) \oplus b_i \\ &= [(b_i + c_i)(1 + b_i + a_i) + (b_i + c_i)(1 + b_i + a_i) + b_i] \bmod 2 \\ &= [5b_i + 2b_i a_i + 2c_i + 2b_i c_i + 2a_i c_i] \bmod 2 \\ &= b_i \bmod 2 = b_i \\ A_i &= p_i \bar{q}_i \oplus c_{i+1} \oplus q_i = b_i \oplus q_i = [b_i + b_i + a_i] \bmod 2 \\ &= [2b_i + a_i] \bmod 2 = a_i \bmod 2 \\ &= a_i \\ C_i &= p_i \bar{q}_i \oplus c_{i+1} \oplus p_i = b_i \oplus p_i = [b_i + b_i + c_i] \bmod 2 \\ &= [2b_i + c_i] \bmod 2 = c_i \bmod 2 \\ &= c_i \end{aligned} \quad (5)$$

We can cascade n pairs of the CGC gates and ICGC gates to compare two n -bit numbers. Such an example is depicted in Fig. 11 with $n = 6$. There is only one ancillary bit c_0 , initialized to 0. The output is x_1 , also initialized to 0. The final result of x_1 is $0 \oplus c_6$. This

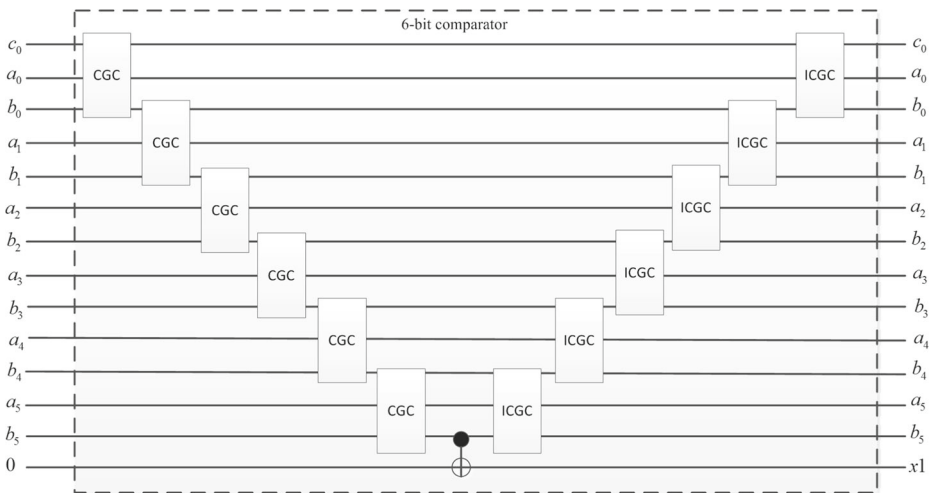


Fig. 11 An example of the comparison of two n -bit numbers for $n = 6$

6-bit comparator needs 6 pairs of the CGC gates and ICGC gates and the quantum cost and quantum delay are both $(12 \times 7 + 1)\Delta$.

3.3 Our Proposed Multi-Bit Comparator

As described in Section 3.2, we conclude that $a \geq b$ if x_1 is 0 and $a < b$ if x_1 is 1. One comparator not only can judge whether a is smaller than b or not, but also can judge whether a equals to b . How to design a reversible multi-bit quantum comparator based on ripple carry idea is an interesting problem.

Considering a case that $a > b$, the result of x_1 only can decide $a \geq b$, by using our proposed n -bit comparator. Then, a and b , swapped with each other, go through the same n -bit comparator and the result is placed into x_2 . As x_2 is 0, it means $b \geq a$. From x_1 , we know $a \geq b$, and we infer $b \geq a$. Finally, we deduce that a equals to b . The whole process is listed in Algorithm 1.

Algorithm 1 The flow-process of our proposed multi-bit comparator

Input: $a = a_{n-1} \dots a_0, b = b_{n-1} \dots b_0, x_1 = 0$ and $x_2 = 0$

Output: $a = a_{n-1} \dots a_0, b = b_{n-1} \dots b_0, x_1$ and x_2 (Begin from left to right)

- 1: Step1: c_{i+1} for $i \geq 0$ is computed from c_1 to c_n
 - 2: Step2: The c_n is written into x_1 by using a NOT gate
 - 3: Step3: c_{i+1} for $i \geq 0$ is restored to its initial state from c_n to c_i
 - 4: Step4: Swap a and b
 - 5: Step5: Again, c_{i+1} for $i \geq 0$ is computed from c_1 to c_n
 - 6: Step6: The c_n is written into x_2 by using a NOT gate
 - 7: Step7: c_{i+1} for $i \geq 0$ is restored to its initial state from c_n to c_i
 - 8: **return** Output x_1 and x_2
-

Our proposed n -bit comparator is depicted in Fig. 12 for $n = 6$. There are three cases for x_1x_2 . If x_1 is 0, it means $a \geq b$ and if x_2 is 0, it means $b \geq a$. So if x_1x_2 is 00, a equals to b . Similarly, if x_1 is 1, it means $a < b$ and x_2 must be 0. If x_2 is 1, it means $b < a$ and x_1 must be 0, too. In this way, we decide whether two n -bit numbers are equal or, otherwise, which of them is the largest by measuring the states of x_1 and x_2 .

4 The Optimized Design of Our Proposed n -bit Comparator

Actually, the design of our proposed n -bit comparator is not optimal. The NOT gates and CNOT gates can be rearranged to minimize the quantum cost and quantum delay of our proposed n -bit comparator. In a sum, we improve the design of our proposed n -bit comparator in four aspects.

- (1) Since there is no carry information ($c_0 = 0$) in the beginning, the comparison of a_0 and b_0 is simplified to a Toffoli gate, indicating $c_0 = 0$ if $a_0 \geq b_0$, and $c_0 = 1$ if $a_0 < b_0$. Similarly, the recovery operation is also simplified to a Toffoli gate.
- (2) The first CNOT gates in all the CGC gates are placed into a vertical line, implying all the CNOT gates can be performed in parallel. Correspondingly, the last CNOT gates in all the ICGC gates also can be performed in parallel. In this way, it will reduce the quantum delay of our proposed n -bit comparator.
- (3) We adjust the order of the Toffoli gate in the current CGC gate and the second CNOT gate in the next CGC gate. The second CNOT gate in the next CGC gate is placed in the same vertical line with the Toffoli gate in the previous CGC gate, implying that the second CNOT gate in the next CGC gate and the Toffoli gate in the previous CGC gate can be performed in parallel. Correspondingly, the first CNOT gate in the previous ICGC gate can also be placed in the same vertical line with the Toffoli gate in the next ICGC gate. The first CNOT gate in the previous ICGC gate and the Toffoli gate in the next ICGC gate can be performed in parallel. As for why the order of the Toffoli

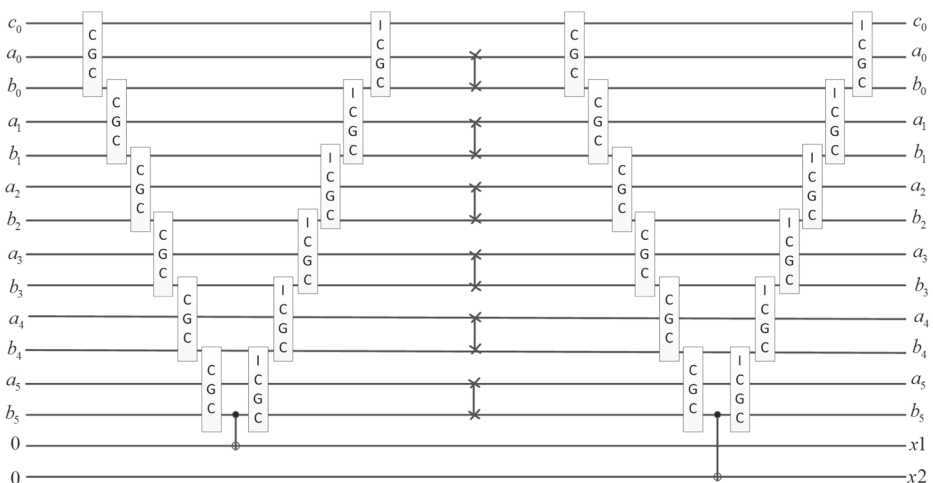


Fig. 12 Our proposed n -bit comparator for $n=6$

gate in the current CGC gate and the second CNOT gate in the next CGC gate can be changed, it will explain in detail in the following step 1 to step 5.

- (4) The last CNOT gate in ICGC gate and the first CNOT gate in Swap gate are cancelled because that two CNOT gates working on a bit consecutively are equal to an identity matrix. That means the state of the bit remains unchanged with or without the two CNOT gates. Subsequently, the last CNOT gate in Swap gate and the following CNOT gate in CGC gate cancel each other.

The optimized version of our proposed n -bit comparator for $n = 6$ is shown in Fig. 13 and the operation details are deduced in the following five steps. Note all the operations are computed from left to right.

- Step 1** c_0, c_1, \dots, c_6 are calculated from c_0 to c_6 and p_1, p_2, \dots, p_6 are also calculated from p_1 to p_6 . The expressions of $c_0, c_1, \dots, c_6, p_1, p_2, \dots, p_6$ are derived in the following (6). From (6), the relation of the c_i for $i = 1, 2, \dots, 5$ in our optimized comparator and the $c_i = CGC(c_{i-1}, a_{i-1}, b_{i-1})$ in the CGC gate is $c_i = c_{i-1} \oplus b_i$. The c_6 and the x_1 satisfy $x_1 = c_6$. That means the carry information is computed from c_1 to c_6 and is written into x_1 by a CNOT gate.
- Step 2** Compared with step 1, step 2 is a reverse process except for missing the last CNOT gates. The expressions of the $C_0, A_0, B_0, \dots, A_5, B_5$ are deduced in (7) from B_5, A_5 to B_0, A_0, C_0 . From these expressions, the states of $C_0, A_0, B_0, \dots, A_5, B_5$ are equal to their states after the first CNOT gates in the vertical line as shown in Fig. 13.
- Step 3** This is a swap process for each pair of A_i and B_i for $i = 0, 1, \dots, 5$. C_1 is already turn back to 0. So we only need verify that the states of each pair of A_i and B_i are swapped with each other. The deduction is detailed in (8). From the deduction, we can conclude that this process can be divided into two sub processes for $i \geq 1$: swap operation and CNOT operation. Each pair of aa_i and bb_i for $i = 1, \dots, 5$ are exactly equivalent to the results that each pair of a_i and b_i swaps with each other

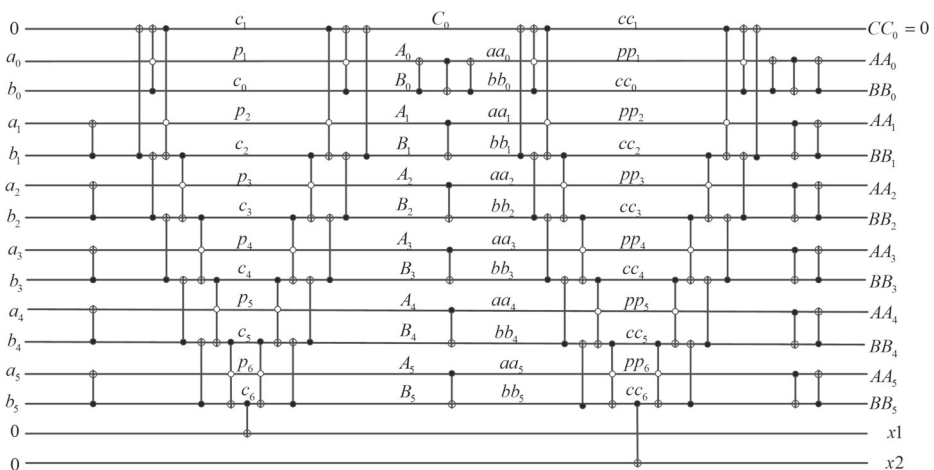


Fig. 13 Our optimized n -bit comparator for $n=6$

and then going through a CNOT gate. Furthermore, step 3 can perform in a time slice.

$$\begin{aligned}
 p_1 &= a_0 \\
 c_0 &= b_0 \\
 c_1 &= b_0 \overline{a_0} \oplus (b_1 \oplus 0) = [b_0(1 + a_0) + b_1] \bmod 2 = b_0 \oplus b_0 a_0 \oplus b_1 \\
 p_2 &= b_1 \oplus a_1 \\
 c_2 &= c_1 \overline{(b_1 \oplus a_1)} \oplus (b_1 \oplus b_2) \\
 p_3 &= b_2 \oplus a_2 \\
 c_3 &= c_2 \overline{(b_2 \oplus a_2)} \oplus (b_2 \oplus b_3) \\
 p_4 &= b_3 \oplus a_3 \\
 c_4 &= c_3 \overline{(b_3 \oplus a_3)} \oplus (b_3 \oplus b_4) \\
 p_5 &= b_4 \oplus a_4 \\
 c_5 &= c_4 \overline{(b_4 \oplus a_4)} \oplus (b_4 \oplus b_5) \\
 p_6 &= b_5 \oplus a_5 \\
 c_6 &= c_5 \overline{(b_5 \oplus a_5)} \oplus b_5
 \end{aligned} \tag{6}$$

Step 4 This step is similar to step 1. The calculation is successively performed from cc_1 to cc_6 and pp_1 to pp_6 . The detailed derivation is listed in (9). If we swap each a_i with b_i in (6), the exchange is exactly expressions in (9). As detailed in step 1, the (6) can judge whether a is equal or greater than b. So the (9) can judge whether b is equal or greater than a and the comparison result is written into x_2 by using a CNOT gate. By identifying the states of $x_1 x_2$, we decide whether two n-bit numbers are equal or, otherwise, which of them is the largest.

Step 5 This step is a reverse of step 4, similar to step 2. The final results are verified in (10). From the (10), the states of $CC_0, AA_0, AA_1, \dots, AA_5$ and BB_0, BB_1, \dots, BB_5 are changed into their initials. This proves that our optimized comparator not only can decide whether two n-bit numbers are equal or, otherwise, which of them is the largest, but also can hold the states unchanged.

From step 1 to step 5, we conclude that our optimized comparator has the same function to compare two n-bit numbers with less quantum cost and quantum delay.

$$\begin{aligned}
 B_5 &= c_5 \overline{p_6} \oplus c_6 = c_5 \overline{p_6} \oplus c_5 \overline{(b_5 \oplus a_5)} \oplus b_5 = [2c_5(1 + b_5 + a_5) + b_5] \bmod 2 = b_5 \\
 A_5 &= p_6 = b_5 \oplus a_5 \\
 B_4 &= c_4 \overline{p_5} \oplus c_5 \oplus B_5 = [2c_4(1 + b_4 + a_4) + b_4 + 2b_5] \bmod 2 = b_4 \\
 A_4 &= p_5 = b_4 \oplus a_4 \\
 B_3 &= c_3 \overline{p_4} \oplus c_4 \oplus B_4 = [2c_3(1 + b_3 + a_3) + b_3 + 2b_4] \bmod 2 = b_3 \\
 A_3 &= p_4 = b_3 \oplus a_3 \\
 B_2 &= c_2 \overline{p_3} \oplus c_3 \oplus B_3 = [2c_2(1 + b_2 + a_2) + b_2 + 2b_3] \bmod 2 = b_2 \\
 A_2 &= p_3 = b_2 \oplus a_2 \\
 B_1 &= c_1 \overline{p_2} \oplus c_2 \oplus B_2 = [2c_1(1 + b_1 + a_1) + b_1 + 2b_2] \bmod 2 = b_1 \\
 A_1 &= p_2 = b_1 \oplus a_1 \\
 B_0 &= c_0 = b_0 \\
 A_0 &= p_1 = a_0 \\
 C_1 &= \overline{a_0} b_0 \oplus c_1 \oplus B_1 = [2\overline{a_0} b_0 + 2b_1] \bmod 2 = 0
 \end{aligned} \tag{7}$$

$$\begin{aligned}
aa_0 &= B_0 \oplus A_0 \oplus B_0 \oplus B_0 \oplus A_0 = [3B_0 + 2A_0] \bmod 2 = B_0 \bmod 2 = B_0 = b_0 \\
bb_0 &= B_0 \oplus A_0 \oplus B_0 = [2B_0 + A_0] \bmod 2 = A_0 \bmod 2 = A_0 = a_0 \\
aa_1 &= A_1 = b_1 \oplus a_1 \\
bb_1 &= A_1 \oplus B_1 = [2b_1 + a_1] \bmod 2 = a_1 \\
aa_2 &= A_2 = b_2 \oplus a_2 \\
bb_2 &= A_2 \oplus B_2 = [2b_2 + a_2] \bmod 2 = a_2 \\
aa_3 &= A_3 = b_3 \oplus a_3 \\
bb_3 &= A_3 \oplus B_3 = [2b_3 + a_3] \bmod 2 = a_3 \\
aa_4 &= A_4 = b_4 \oplus a_4 \\
bb_4 &= A_4 \oplus B_4 = [2b_4 + a_4] \bmod 2 = a_4 \\
aa_5 &= A_5 = b_5 \oplus a_5 \\
bb_5 &= A_5 \oplus B_5 = [2b_5 + a_5] \bmod 2 = a_5
\end{aligned} \tag{8}$$

$$\begin{aligned}
pp_1 &= aa_0 = b_0 \\
cc_0 &= bb_0 = a_0 \\
cc_1 &= bb_0 \overline{aa_0} \oplus (bb_1 \oplus 0) = a_0 \overline{b_0} \oplus a_1 \\
pp_2 &= aa_1 = b_1 \oplus a_1 \\
cc_2 &= bb_2 \oplus bb_1 \oplus cc_1 \overline{aa_1} = cc_1 (\overline{b_1 \oplus a_1}) \oplus a_2 \oplus a_1 \\
pp_3 &= aa_2 = b_2 \oplus a_2 \\
cc_3 &= cc_2 (\overline{pp_3}) \oplus (bb_2 \oplus bb_3) = cc_2 (\overline{b_2 \oplus a_2}) \oplus (a_2 \oplus a_3) \\
pp_4 &= aa_3 = b_3 \oplus a_3 \\
cc_4 &= cc_3 (\overline{pp_4}) \oplus (bb_3 \oplus bb_4) = cc_3 (\overline{b_3 \oplus a_3}) \oplus (a_3 \oplus a_4) \\
pp_5 &= aa_4 = b_4 \oplus a_4 \\
cc_5 &= cc_4 (\overline{pp_5}) \oplus (bb_4 \oplus bb_5) = cc_4 (\overline{b_4 \oplus a_4}) \oplus (a_4 \oplus a_5) \\
pp_6 &= aa_5 = b_5 \oplus a_5 \\
cc_6 &= cc_5 (\overline{pp_6}) \oplus bb_5 = cc_5 (\overline{b_5 \oplus a_5}) \oplus a_5
\end{aligned} \tag{9}$$

$$\begin{aligned}
BB_5 &= cc_5 \overline{pp_6} \oplus cc_6 \oplus pp_6 = [2cc_5 \overline{pp_6} + bb_5 + pp_6] \bmod 2 = [a_5 + b_5 + a_5] \bmod 2 = b_5 \\
AA_5 &= BB_5 \oplus pp_6 = b_5 \oplus b_5 \oplus a_5 = a_5 \\
BB_4 &= cc_4 \overline{pp_5} \oplus cc_5 \oplus a_5 \oplus pp_5 = [2cc_4 \overline{pp_5} + 2a_4 + 2a_5 + b_4] \bmod 2 = b_4 \\
AA_4 &= BB_4 \oplus pp_5 = b_4 \oplus b_4 \oplus a_4 = a_4 \\
BB_3 &= cc_3 \overline{pp_4} \oplus cc_4 \oplus a_4 \oplus pp_4 = [2cc_3 \overline{pp_4} + 2a_3 + 2a_4 + b_3] \bmod 2 = b_3 \\
AA_3 &= BB_3 \oplus pp_4 = b_3 \oplus b_3 \oplus a_3 = a_3 \\
BB_2 &= cc_2 \overline{pp_3} \oplus cc_3 \oplus a_3 \oplus pp_3 = [2cc_2 \overline{pp_3} + 2a_2 + 2a_3 + b_2] \bmod 2 = b_2 \\
AA_2 &= BB_2 \oplus pp_3 = b_2 \oplus b_2 \oplus a_2 = a_2 \\
BB_1 &= cc_1 \overline{pp_2} \oplus cc_2 \oplus a_2 \oplus pp_2 = [2cc_1 \overline{pp_2} + 2a_1 + 2a_2 + b_1] \bmod 2 = b_1 \\
AA_1 &= BB_1 \oplus pp_2 = b_1 \oplus b_1 \oplus a_1 = a_1 \\
BB_0 &= cc_0 \oplus pp_1 \oplus cc_0 = pp_1 = b_0 \\
AA_0 &= cc_0 \oplus pp_1 \oplus BB_0 = a_0 \oplus b_0 \oplus b_0 = a_0 \\
CC_0 &= cc_0 \overline{pp_1} \oplus cc_1 = a_0 \overline{b_0} \oplus a_0 \overline{b_0} \oplus a_1 \oplus a_1 = 0
\end{aligned} \tag{10}$$

5 Complexity Analysis

We analyze the complexity of our proposed n -bit comparator and its optimized version in terms of quantum cost, quantum delay and ancillary bits.

The quantum cost is given by (11) and the quantum delay is given by (12) for our proposed n -bit comparator ($n \geq 1$), named as our comparator. The quantum cost is given by

(13) and the quantum delay is given by (14) for our optimized version of our n -bit comparator, named as our optimized comparator. For example, Figs. 13 and 14 give the 6-bit comparator and its optimized version, which can compare two numbers less than 63. The quantum cost and delay of the 6-bit comparator and its optimized version are 188Δ , 188Δ , 168Δ and 133Δ , respectively.

$$OC = (2 + 7 \times 4n + 3n)\Delta \quad (11)$$

$$OD = (2 + 7 \times 4n + 3n)\Delta \quad (12)$$

$$OPC = (28n)\Delta \quad (13)$$

$$OPD = \begin{cases} 28n\Delta, & n = 1 \\ 53n\Delta, & n = 2 \\ (53 + 20(n - 2))\Delta, & n \geq 2 \end{cases} \quad (14)$$

Figure 14 shows the quantum costs of our proposed method versus the existing comparators, including *method – 1* presented by Wang et al., *method – 2* proposed by Al-Rabadi, *method – 3* presented by Thapliyal et al., *method – 4* proposed by Vudadha et al. and our proposed method. The total quantum costs for an n -bit comparator performed by different comparators are given as follows:

Method – 1: The quantum cost is $O(n^2)$.

Method – 2: The quantum cost is $(39 \times n + 9)\Delta$.

Method – 3: The quantum cost is $(18 \times n + 9)\Delta$.

Method – 4: The quantum cost is $(14 \times n)\Delta$.

Our comparator: The quantum cost is $(2 + 31n)\Delta$.

Our optimized comparator: $(28n)\Delta$ for our optimized comparator.

From Fig. 14, it can be seen that the quantum cost of *method – 1* increases sharply compared with the quantum costs of the other five comparators. The quantum cost of

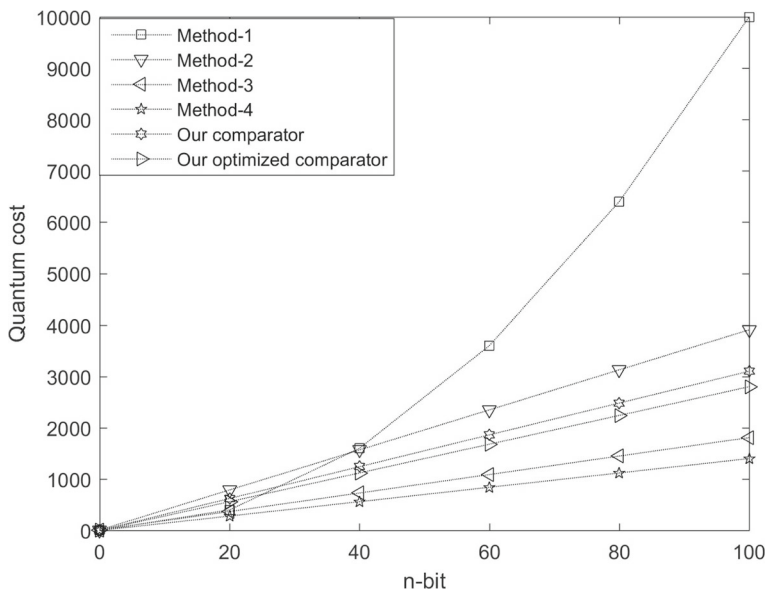


Fig. 14 The quantum costs of our method and four existing methods

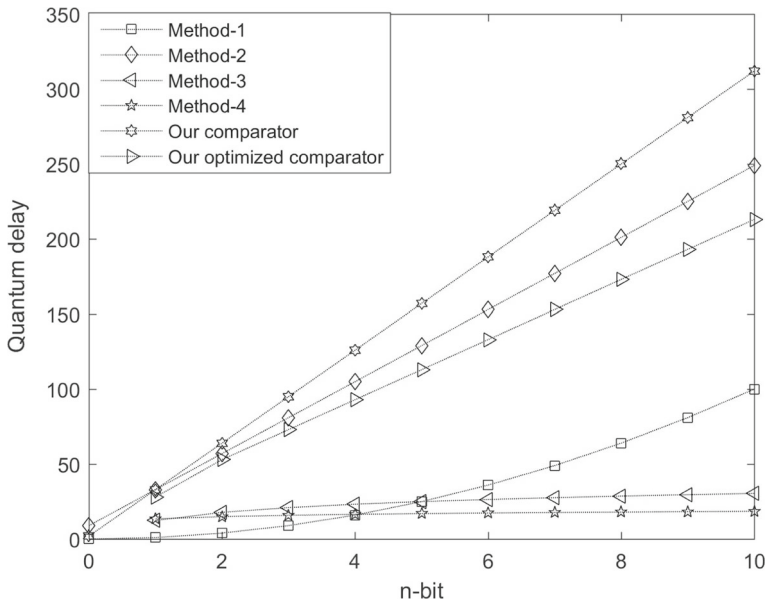


Fig. 15 The quantum costs of our method and four existing methods

Ouroptimizedcomparator is lower than the quantum cost of *Ourcomparator*, but it is higher than the quantum costs of *method – 3* and *method – 4*. Fortunately, the quantum cost of *Ouroptimizedcomparator* increases linearly with the increase of n .

Figure 15 shows the quantum delay of our proposed method versus the existing four comparators. The total quantum delays for an n -bit comparator performed by different comparators are given as follows:

- Method – 1*: The quantum delay is $O(n^2)$.
- Method – 2*: The quantum delay is $24 \times n + 9$.
- Method – 3*: The quantum delay is $18 \times \log(2 \times n) + 7$.
- Method – 4*: The quantum delay is $5 \times \log(2 \times n) + 12$.
- Ourcomparator*: The quantum cost is $(31 \times n + 2)$.
- Ouroptimizedcomparator*: Shown in (13).

From Fig. 15, it can be seen that the quantum delay of *method – 1* increases sharply with the increase of n . Our proposed comparator has a lower quantum delay than the quantum delays in *method – 1* and *method – 2*. The quantum delay of our proposed method increase linearly with the increase of n while the quantum delays in *method – 3* and *method – 4* only increase logarithmically. That is because *method – 3* and *method – 4* realize the comparison of two n -bit binary numbers by combing the 1-bit comparators or the 2-bit comparators in tree structure, while *method – 1*, *method – 2* and our proposed method realize the comparison by cascading the 1-bit comparators in a serial way. Although the tree-based structure has advantages in quantum delay over the serial-based structure, this tree-based structure generally introduce more auxiliary bits than the serial-based structure.

Figure 16 shows the auxiliary bits of our proposed method versus the existing four comparators. The total auxiliary bits for an n -bit comparator performed by different comparators are given as follows:

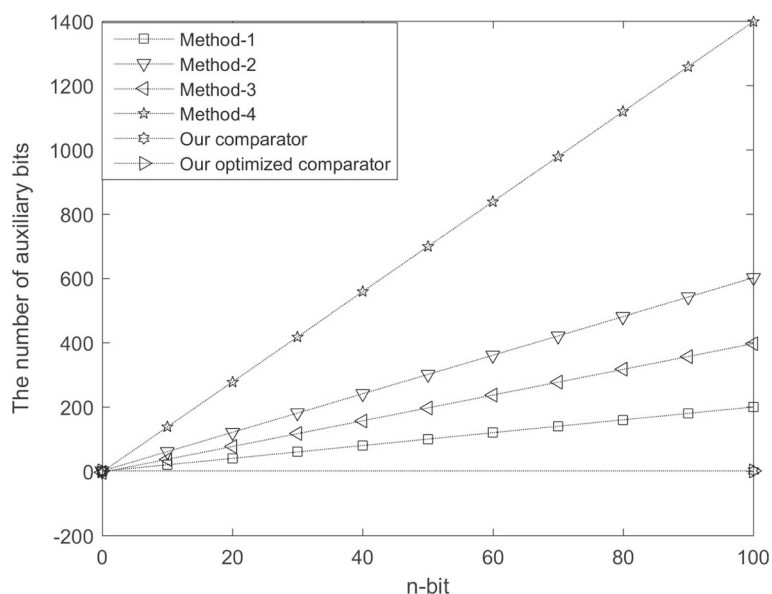


Fig. 16 The number of auxiliary bits of our method and four existing methods

Method – 1: The number of auxiliary bits is $2 \times n$.

Method – 2: The number of auxiliary bits is $6 \times n + 1$.

Method – 3: The number of auxiliary bits is $4 \times n - 3$.

Method – 4: The number of auxiliary bits is $4 \times n - 2$.

Our comparator: The number of auxiliary bits is 1.

Our optimized comparator: 1 auxiliary bit for our optimized comparator.

From Fig. 16, we can see that the number of auxiliary bits of the comparators except our proposed method increase linearly with the increase of n . Therefore the number of auxiliary bits of our method is negligible when compared to the existing ones. That is an important advantage over the existing ones.

6 Conclusions

This paper presents an efficient design of reversible multi-bit quantum comparator via only a single ancillary bit. Our proposed multi-bit quantum comparator not only can compare more bits with only one auxiliary bit in less quantum cost compared with the existing quantum logic comparators, but also can hold the quantum logic states unchanged. Furthermore, we optimize the design of our proposed comparator in quantum cost and quantum delay. But the quantum delay and quantum cost of our optimized comparator are still inferior to the quantum logic comparators in tree-based structure. A future work direction is to further optimize the design of our proposed comparator in quantum cost and quantum delay.

Acknowledgements This work is supported by the National Natural Science Foundation of China (No.61762014, No.61462026 and No.61762012), the Opening Project of Guangxi Colleges and Universities Key Laboratory of robot & welding (Guilin University of Aerospace Technology), the Opening Project of Shaanxi Key Laboratory of Complex Control System and Intelligent Information Processing, and the

Research Fund of Guangxi Key Lab of intelligent integrated automation. This work is also partly supported by the Project of Science and Technology of Jiangxi province (No. 20161BAB202065).

References

1. Jiang, N., Dang, Y., Wang, J.: Quantum image matching. *Quantum Inf. Process.* **15**(9), 3543–3572 (2016)
2. Victor, P., Nelson, N.: *Digital logic circuit analysis and design*. Prentice Hall, Upper Saddle River (1995)
3. Al-rabadi, A.N.: Closed-system quantum logic network implementation of the viterbi algorithm. *Facta Univ.-Ser.: Elec. Energ.* **22**(1), 1–33 (2009)
4. Wang, D., Liu, Z., Zhu, W., Li, S.: Design of quantum comparator based on extended general Toffoli gates with multiple targets. *Comput. Sci.* **39**(9), 302–306 (2012)
5. Thapliyal, H.N.: Design of reversible sequential circuits optimizing quantum cost, delay, and garbage outputs. *Acm Journal on Emerging Technologies in Computing Systems* **6**(4), 14–45 (2010)
6. Thapliyal, H., Ranganathan, N., Ferreira, R.: Design of a comparator tree based on reversible logic. In: 2010 10th IEEE Conference on Nanotechnology (IEEE-NANO), pp. 111–1116 (2010)
7. Vudadha, C., Phaneendra, P.S., Sreehari, V., Ahmed, S.E., Muthukrishnan, N.M., Srinivas, M.B.: Design of Prefix-Based optimal reversible comparator. In: 2012 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 201–206 (2012)
8. Schumacher, B.: Quantum coding. *Phys. Rev. A* **51**(4), 2738–2747 (1995)
9. Dirac, P.A.M.: *The principles of quantum mechanics*. Oxford University Press, Oxford (1947)
10. Li, H.-S., Qingxin, Z., Zhou, R.-G., Li, M.-C., Song, L.: Multidimensional color image storage, retrieval, and compression based on quantum amplitudes and phases. *Inform. Sci.* **273**, 212–232 (2014)
11. Li, H.-S., Qingxin, Z., Lan, S., Wu, Q.: The quantum search algorithms for all solutions. *Int. J. Theor. Phys.* **52**(6), 1893–1907 (2013)
12. Li, H.-S., Qingxin, Z., Zhou, R.-G., Song, L., Yang, X.-j.: Multi-dimensional color image storage and retrieval for a normal arbitrary quantum superposition state. *Quantum Inf. Process.* **13**(4), 991–1011 (2014)
13. Zhou, R., Shi, Y., Zhang, M., et al.: A novel reversible ZS gate and its application for optimization of quantum adder circuits. *J. Circ. Syst. Comput.* **20**(06), 1107–1129 (2011)
14. Cheng, K.W., Tseng, C.C.: Quantum full adder and subtractor. *Electron. Lett.* **38**(22), 1343–1344 (2002)
15. Cuccaro, S.A., Draper, T.G., Kutin, S.A., et al.: A new quantum ripple-carry addition circuit. [arXiv:quant-ph/0410184](https://arxiv.org/abs/quant-ph/0410184) (2004)
16. Draper, T.G., Kutin, S.A., Rains, E.M., et al.: A logarithmic-depth quantum carry-lookahead adder. [arXiv:quant-ph/0406142](https://arxiv.org/abs/quant-ph/0406142) (2004)