Article

Efficient Designs of Quantum Adder/Subtractor Using Universal Reversible Gate on IBM Q

Mohamed Osman 1,2, * and Khaled El-Wazan 2,3

1 Department of Mathematics, Faculty of Science, Damanhour University, Damanhour 22511, Egypt
2 Academy of Scientific Research and Technology (ASRT), Cairo 11516, Egypt; khaled_elwazan@alex-sci.edu.eg
3 Department of Mathematics and Computer Science, Faculty of Science, Alexandria University, Alexandria 21568, Egypt
* Correspondence: mmoneim@sci.dmu.edu.eg

Abstract: Reversible arithmetic and logic unit (ALU) is a necessary part of quantum computing. In this work, we present improved designs of reversible half and full addition and subtraction circuits. The proposed designs are based on a universal one type gate (G gate library). The G gate library can generate all possible permutations of the symmetric group. The presented designs are multi-function circuits that are capable of performing additional logical operations. We achieve a reduction in the quantum cost, gate count, number of constant inputs, and delay with zero garbage, compared to relevant results obtained by others. The experimental results using IBM Quantum Experience (IBM Q) illustrate the success probability of the proposed designs.

Keywords: reversible gate; quantum adder; quantum circuit; quantum cost; quantum processor; quantum subtractor

1. Introduction

Reversible logic [1,2] is an essential part of building a reversible circuit. A circuit is called reversible if the circuit maps each input to an exclusive output, and the output contains enough information to retrieve the input, i.e., there is a one-to-one consistency among the input/output [3]. Many applications have been proposed for reversible circuits in many technologies [3], e.g., nano-technologies, low-power CMOS [4,5], quantum-dot cellular automata [6,7].

Quantum computers [8–10] are constructed of reversible circuits, which have demonstrated capabilities for solving certain problems faster than classical counterparts [11], e.g., Grover presented a quantum algorithm for searching in an unstructured database with quadratic speed-up over classical computers [12]. Shor provided a polynomial-time quantum algorithm for factoring integers into their prime factors [13]. The reversible nature of quantum computers and quantum logic gates [14] demands the need to construct efficient reversible circuits, taking into consideration certain parameters [15], e.g., number of garbage output, quantum cost, number of constant inputs, gate count and delay.

Arithmetic and logic unit (ALU) [16] is an essential part of any computational device. It works as a multi-function circuit capable of performing a predefined set of logical and arithmetic operations. Two of the fundamental functions of ALU are arithmetic addition and subtraction. Several designs have been proposed for full adder and full subtractor reversible circuits. For instance, Thapliyal and Ranganathan proposed a full subtractor using two TRG gates [17]. Gupta et al. introduced an improved construction of full reversible adder/subtractor [18] using three F2G gates and a single MUX gate. Moghimi and Reshadinezhad provided a 4 × 4 full adder and subtractor [19] using one Peres gate and one Fredkin gate with a reduction in cost. Montaser et al. presented a full adder and subtractor design using two R-gate [20].
Recently, many reversible \( n \)-bit gate libraries have been proposed [21–23]. These libraries are proven to be single-type gate libraries, universal for reversible circuit synthesis, and extendable according to the number of bits in reversible circuit design. The importance of using single-type gate library stems from the need to build cheaper reversible circuits by using similar building blocks [16].

In this paper, we propose improved designs of half and full adder/subtractor circuits. The proposed designs are based on the \( G^3 \) gate library [21] which, in turn, is capable of generating the permutation group of size 40,320 (\( 2^{31} \)). This permutation group is also a subset of the symmetric group of degree 3. The introduced designs are capable of performing the addition and subtraction operations as part of ALU. The suggested designs are found to be of low quantum cost, delay, gate count, constant input, and zero garbage outputs when compared to relevant work. Furthermore, the proposed designs are implemented and tested on IBM Q [24] using Python SDK [25] to measure the success probability of obtaining the correct results.

2. Preliminaries

2.1. Basic Definitions

**Definition 1.** Given two bits \( X \) and \( Y \), a half adder (HA) circuit is a combinational circuit which is capable of performing the arithmetic addition of \( X \) and \( Y \) [26]. The output of the half adder circuit can be expressed in sum-of-products (SOP) form as follows [26]:

\[
\text{Sum} = X \oplus Y \\
C_{\text{out}} = XY,
\]

(1)

where \( C_{\text{out}} \) is the carry from the arithmetic addition of \( X \) and \( Y \), and \( \oplus \) is the exclusive-or (XOR) operation.

**Definition 2.** Given three bits \( X, Y \) and \( C_{\text{in}} \), a full adder (FA) circuit is a compositional circuit that fulfills the arithmetic addition of \( X, Y \) and \( C \), where \( X \) and \( Y \) are the basic inputs, and \( C_{\text{in}} \) is the carry from the last addition operation of the prior least significant bits [26]. The output of the full adder circuit can be expressed in SOP as follows [26,27]:

\[
\text{Sum} = X \oplus Y \oplus C_{\text{in}} \\
C_{\text{out}} = XY \oplus XC_{\text{in}} \oplus YC_{\text{in}},
\]

(2)

where \( C_{\text{out}} \) is the carry from the arithmetic addition of \( X, Y \) and \( C_{\text{in}} \).

**Definition 3.** A half subtractor (HS) circuit is a combinational circuit that calculates the arithmetic subtraction of two input bits [26]. For two input bits \( X \) and \( Y \), the output of the half subtractor can be expressed in SOP as follows:

\[
\text{Diff} = X \oplus Y \\
B_{\text{out}} = XY \oplus Y,
\]

(3)

where \( \text{Diff} \) is the difference and \( B_{\text{out}} \) is the borrow out.

**Definition 4.** A full subtractor (FS) circuit is a compositional circuit that fulfills the arithmetic subtraction of three input bits [26]. For three input bits \( X, Y \) and \( B_{\text{in}} \), the output of the full subtractor can be expressed in SOP as follows:

\[
\text{Diff} = X \oplus Y \oplus B_{\text{in}} \\
B_{\text{out}} = XY \oplus XB_{\text{in}} \oplus YB_{\text{in}} \oplus Y \oplus B_{\text{in}},
\]

(4)

**Definition 5.** A Boolean function \( f(x_1, x_2, \ldots, x_n) \) with \( n \) input variables and \( n \) output variables is a controlled mapping from \( \{0, 1\}^n \) to \( \{0, 1\}^n \), i.e., \( f : \{0, 1\}^n \to \{0, 1\}^n \).
Definition 6. The truth table of a Boolean function $f$ that maps an input vector $v_i \in \{0, 1\}^n$ to $\{0, 1\}^n$, is defined as the lexicographical order of $v_i$ mapped to $f(v_i) \in \{0, 1\}^n$:

\[
\begin{align*}
v_1 &= (0, 0, \cdots, 0) \rightarrow f(v_1) \\
v_2 &= (0, 0, \cdots, 1) \rightarrow f(v_2) \\
&\vdots \\
v_{2^n} &= (1, 1, \cdots, 1) \rightarrow f(v_{2^n}).
\end{align*}
\]

Definition 7. Given a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$, $f$ is called reversible if and only if each input vector $v_i \in \{0, 1\}^n$ is mapped to one and only one exclusive output $f(v_i)$. 

Definition 8. We say that a gate with $n$ input and $n$ output ($n \times n$) is reversible, if that gate is a realization of a reversible Boolean function.

Definition 9. Given a set of reversible gates $L$, we say that $L$ is a library, if $L$ can be used to build reversible circuits [21,22].

Definition 10. Given a reversible gate library $L$, we say that $L$ is universal, if $L$ can be used to synthesize any reversible circuit ($n \times n$) [21,22].

Definition 11. Given a $(k \times k)$ reversible gate $Z$ acting on $(n \times n)$ reversible circuit, the Z gate is denoted as $Z^n_{p_1,p_2,\ldots,p_k}$, where the superscript represents the number of bit wires in the reversible circuit and the subscript taking $p_1, p_2, \ldots, p_{k-1}$ as the indices of the control bits and $p_k$ as the index of the target bit, as shown in Figure 1.

![Figure 1](image-url)

**Figure 1.** A general representation of a reversible $(k \times k)$ gate with controls $p_1, p_2, \ldots, p_{k-1}$ control wires and $p_k$ target wire.

2.1.1. NOT Gate

The NOT ($N$) gate is a Boolean $(1 \times 1)$ reversible gate that works on a single bit and inverts it unconditionally. It has quantum cost of zero [28]. Figure 2 illustrates all possible variations of $N$ gate working on 3-bit circuit. The effect of the $N$ gate acting on a bit indexed $l \in A$ in a circuit of $n$ bits is demonstrated as follows:

\[
N^m_l : y_l = x_l \oplus 1,
\]

where $\oplus$ is the exclusive-or operation (XOR).

![Figure 2](image-url)

**Figure 2.** All possible variations of $N$ gate acting on three bits.
2.1.2. Feynman Gate

The Feynman (C) gate is a reversible Boolean (2×2) gate (also known as the Controlled-NOT gate) that acts on two input bits and inverts the target bit if and only if the control bit is set to one. The quantum cost of the C gate is one [28]. Figure 3 illustrates the six possible variations of C gate.

The effect of the C gate acting on wires indexed \( i, j \in A \) is described as follows:

\[
C_{ij}^n : y_i = x_i \\
y_j = x_i \oplus x_j.
\]

(7)

![Figure 3](https://example.com/figure3.png)

**Figure 3.** All possible variations of \( C \) gate acting on three bits.

2.1.3. Controlled Square-Root NOT Gate

The controlled square-root NOT gate is a reversible (2×2) gate that acts on two input bits, and apply the square-root NOT gate on the target bit if and only if the control bit is set to 1. There are two possible gates for the controlled square-root NOT gate: controlled-V(\( \nu \)) and controlled-\( V^{-1}(u) \) gates. Figure 4 illustrates the possible variations of \( \nu \) and \( u \) gates acting on three qubits reversible circuit.

The \( \nu \) and \( u \) gates acting on wires indexed \( g, h \in A \) have the following properties:

\[
\nu_{gh}^u \cdot u_{gh}^u = u_{gh}^u \cdot \nu_{gh}^u = I, \\
u_{gh}^u \cdot u_{gh}^n = u_{gh}^n \cdot \nu_{gh}^u = C_{gh}^n.
\]

(8)

where \( I \) represents the identity gate.

![Figure 4](https://example.com/figure4.png)

**Figure 4.** All possible variations of \( \nu \) and \( u \) gate acting on three qubits circuit.

2.1.4. Toffoli Gate

The Toffoli T gate [29] is a reversible Boolean (3×3) gate that takes three input bits and produce an output by flipping the target bit if and only if the control bits are set to one, as illustrated in Figure 5b. The T gate can be decomposed to five (2×2) elementary gates [30,31], as illustrated in Figure 5b, and thus it has a quantum cost of five. The action of the T gate acting on wires indexed \( r, s, t \in A \) is described as follows:

\[
T_{rst} : y_r = x_r \\
y_s = x_s \\
y_t = x_t \oplus x_r x_s.
\]

(9)
The gate representation of Figure 6. The gate representation of $G^3$ gates for a 3-bit reversible circuit [21].

Figure 5. The $T^3$ gate where: (a) all possible variations of $T$ gate acting on three circuit, (b) the $T_{123}$ decomposition into five elementary gates.

The $T$ gate can be generalized to work in a circuit of $n$ bits as a $(k \times k)$ gate, taking $k$-inputs and producing $k$-outputs flipping the target bit if and only if the $k - 1$ control bits are set to 1. The $T$ gate acting on wires indexed $p_1, p_2 \cdots p_{k-1}, p_k \in A$ can be shown as follows:

$$
T_{p_1p_2 \cdots p_k} : 
\begin{align*}
y_{p_1} &= x_{p_1} \\
y_{p_2} &= x_{p_2} \\
&\vdots \\
y_{p_k} &= x_{p_k} \oplus \prod_{q=1}^{k-1} x_{p_q},
\end{align*}
$$

where $\prod_{q=1}^{k-1} x_{p_q}$ is a product term of $k - 1$ variables.

2.1.5. G Gate

The $G^a$ gate library was introduced by Younes [21]. It is a reversible universal gate library that can be used to synthesize any reversible circuit with $n$-input/output variables, for $n \geq 2$. The $G^a$ associates $N^a$ gate, $C^a$ gate and the $T^a$ gate to build a library that acts on $n$-bits circuits. Figure 6 illustrates the $G^3$ library in a circuit of three bits. The action of the $G$ gate acting on three bits indexed $a, b, c \in A$ in a circuit of $n = 3$ wires can be described as follows:

$$
G_{abc}^3 : 
\begin{align*}
y_a &= x_a \oplus 1, \\
y_b &= x_a \oplus x_b, \\
y_c &= x_c \oplus x_a x_b.
\end{align*}
$$

Figure 6. The gate representation of $G^3$ gates for a 3-bit reversible circuit [21].
Definition 12. A garbage output [15] is any meaningless output of a reversible quantum circuit.

Definition 13. Given a reversible quantum circuit, the quantum cost (QC) [15] is defined as the number of elementary quantum gates used to construct this circuit.

There are two cost metrics to calculate the cost of elementary quantum gates: $cost_{11}$ metric [30,32,33] and $cost_{01}$ metric [34]. The $cost_{11}$ metric regards the QC of $(1 \times 1)$ and $(2 \times 2)$ gates as one, respectively. However, the $cost_{01}$ metric regards the QC of $(1 \times 1)$ and $(2 \times 2)$ gates as 0 and 1 [28], respectively. In this paper, $cost_{01}$ metric is used to calculate the QC.

Definition 14. Gate count (GC) [15] of a reversible quantum circuit is the number of quantum gates used to construct this circuit.

Definition 15. Given a reversible quantum circuit, a constant input (CI) [15] is an input bit that is set with a constant input value, and it is used to compute a certain Boolean function.

Definition 16. Given a logic circuit, the amount of time for a signal to circulate from the input to output is called delay [26].

It is known that the delay in reversible quantum circuits is dependent on the technologies invested in producing those circuits, and it can be explicitly determined when the technologies used in production is determined [15].

The method introduced in [15] for calculating the delay in quantum circuits is defined as follows:

1. Every $(1 \times 1)$ or $(2 \times 2)$ quantum gates will have only one unit delay.
2. Every quantum gates with $(3 \times 3)$ or bigger will be substituted with their equivalent $(1 \times 1)$ gates and $(2 \times 2)$ gates.
3. Find the path from input to output which has the maximum delay; this is considered the unit delay $\Delta$ of the circuit.

3. The Proposed Designs

In this section, we introduce the proposed designs for the reversible HA/HS circuits and the reversible FA/FS circuits.

3.1. The Proposed Reversible Half Adder/Subtractor Circuits

We construct our proposed HA/HS using a single $G_{123}^1$ gate, as shown in Figure 7a. The proposed design is a $3 \times 3$ circuit, where $x_3$ is the control bit. Setting $x_1 = X$, $x_2 = Y$ and $x_3 = 0$, the proposed design will work as a HA such that the bit $y_2$ will output $Sum$, and $y_3$ will output $C_{out}$; the $y_1$ bit will produce $X$ which is the negation (NOT) operation. Setting $x_1 = X$ and $x_2 = x_3 = Y$, the proposed design will work as a HS, where the bit $y_2$ will output $Diff$ and the bit $y_3$ will output $B_{out}$; the bit $y_1$ will perform the NOT operation on $X$. Figure 7b shows the Toffoli decomposition for the proposed HA/HS. Figure 7c illustrates the equivalent decomposition using five elementary gates, where QC equals to four, and the delay equals to three. Table 1 summarizes the possible operations that the proposed HA/HS is able to perform.
Figure 7. (a) The proposed HA/HS using the $G^3$ library, (b) the Toffoli decomposition of the proposed HA/HS, (c) the optimized decomposition of the proposed HA/HS into five elementary gates with QC equal to four.

Table 1. The possible functions that can be performed using the proposed design Section 3.1.

|   | $x_1$ | $x_2$ | $x_3$ | $y_1$ | $y_2$ | $y_3$ | Result |
|---|---|---|---|---|---|---|---|
| Proposed design Section 3.1 | $X$ | $Y$ | 0 | $\overline{X}$ | Sum | $C_{out}$ | ADD, NOT |
|   | $X$ | $Y$ | $Y$ | $\overline{X}$ | Diff | $B_{out}$ | SUB, NOT |

3.2. The Proposed Reversible Full Adder Circuit

Using two $G^3$ gates, we are able to construct a FA circuit capable of performing the addition operation. We use both $G^3_{123}$ and $G^3_{231}$ cascaded as illustrated in Figure 8a. Setting $x_2 = 0$ as a constant bit and considering $x_1$, $x_3$ and $x_4$ as $X$, $Y$ and $C_{in}$ respectively, the proposed FA will calculate the addition operation where the $y_2$ bit will output the $\text{Sum}$ and $y_3$ bit will output the $C_{out}$; in addition, $y_1$ will output $\overline{X}$, and $y_3$ will output $\overline{X} \oplus \overline{Y}$ which is the exclusive-NOR (XNOR) operation. It is clear that the proposed FA does not produce any garbage output and has only one constant bit. By changing the values of $x_2$ and $x_4$, the proposed FA can perform additional logical operations on $x_1$ and $x_3$ such as NOT, AND, exclusive-or (XOR), OR, NOR and NAND, also with no garbage output. Table 2 summarizes the possible operations that the proposed FA is able to perform.

Figure 8b illustrates the equivalent decomposition for the proposed FA circuit into its Toffoli decomposition. Using the rules of optimization defined in [30] and a Toffoli decomposition technique in [31], the number of elementary gates used to build the proposed FA can be reduced from 14 gates to eight gates, as shown in Figure 8c. Figure 8d shows the optimized decomposition of the proposed FA into eight elementary gates, with a QC equals to six (using cost01 metric) and a delay equals to $4\Delta$. 
3.3. The Proposed Reversible Full Subtractor Circuit

We are able to build a FS that is capable of performing the binary subtraction operation, using two $G^3$ gates. The proposed FS uses $G_{132}^3$ and $G_{231}^3$ in addition to two NOT gates, as illustrated by Figure 9a. Setting $x_2 = x_3 = Y$, and considering $x_1$ and $x_4$ as $X$ and $B_{in}$, respectively, the proposed FS will perform the binary subtraction operation and the $y_2$ bit will output $Diff$ and the $y_4$ bit will output $B_{out}$; in addition, $y_1$ outputs the result of applying the NOT operation on $X$ and $y_3$ performs the XOR operation $X \oplus Y$. It is clear that the proposed FS has no constant inputs and no garbage as well. Table 2 summarizes the possible operations that the proposed FS is capable of.

The proposed FS can be further optimized by replacing the $T^3$ gates with a suitable decomposition, as illustrated in Figure 9b. It is evident that the proposed FA design has a QC of eight and a $7\Delta$ delay.
Proposed design Section 3.2

| x1 | x2 | x3 | y1 | y2 | y3 | y4 | Result          |
|----|----|----|----|----|----|----|----------------|
| X  | 0  | Y  | C\textsubscript{in} |   | Sum | X \oplus Y | C\textsubscript{out} | ADD, NOT, XNOR  |
| X  | 0  | Y  | 0   |   |   | X \oplus Y |   | NOT, AND, XNOR, XOR |
| X  | 0  | Y  | 1   |   |   | X Y \oplus X \oplus Y | X \oplus Y | NOT, OR, XNOR |
| X  | 1  | Y  | 0   |   |   | X Y | X \oplus Y | NOT, NAND, XNOR, XOR |
| X  | 1  | Y  | 1   |   |   | X Y \oplus X \oplus Y | X \oplus Y | NOT, NOR, XNOR, XOR |

Proposed design Section 3.3

| x1 | x2 | y3 | y4 | Result          |
|----|----|----|----|----------------|
| X  | Y  | Y  | B\textsubscript{in} | ADD, NOT, XNOR |
| X  | Y  | 1   |   | NOT, NAND, XNOR, XOR |

Table 2. The possible Boolean operations that can be performed using the proposed designs Sections 3.2 and 3.3 by changing the input bits x\_2, x\_3 and x\_4.

4. Experimental Results and Discussion

In this section, the experiments on IBM Q [24] using Qiskit Python SDK [25] are presented for the proposed designs. Each design is tested 8192 times. In addition, we discuss and compare the results obtained by proposed designs with other results in the relevant work. The main criteria of comparison are QC, GO, CI, GC, and circuit delay.

Executing the Proposed Designs on IBM Q

Figure 10 shows the actual sequence of gates that are used to implement the proposed designs on IBM Q [24]. This involves Hadamard gate (denoted H), \pi/4 gate (denoted T), complex conjugate transpose of T (denoted T\textdagger), rotation gate with three Euler angles (denoted U\_3), and controlled-not gate. All the proposed designs are tested on IBM Q [24] using Qiskit Python SDK [25], each design is tested 8192 times. The success probability is the probability of obtaining the desired output for a given function over all trials. In the presented work, the maximum number of trials permitted by IBM Q 5 Santiago backend specification [24] is 8192 trials and this is the number of trials we used. The average success probability for each proposed design is calculated.

Table 3 shows the truth table for the proposed HA, and illustrates the probability of obtaining the correct output after executing on IBM Q [24], with a maximum success probability 0.8829 and an average success probability 0.8259. Further more, Table 4 shows the truth table of the proposed HS, and illustrates the probability of obtaining the correct output after executing on IBM Q with a maximum success probability 0.7545 and an average success probability 0.7456.

![Figure 9](image_url)

Figure 9. (a) The proposed design of the FS using the G\textsuperscript{3} library, and (b) the optimized decomposition of the proposed FA into 12 basic gates with a total QC equals to eight.
Figure 10. The actual representation for the proposed designs, where (a) The proposed design Section 3.1: HA/HS circuit representation on IBM’s using Qiskit Python SDK, (b) The proposed design Section 3.2: FA circuit representation on IBM’s using Qiskit Python SDK and its decomposition and (c) The proposed design Section 3.3: FS circuit representation on IBM’s using Qiskit Python SDK and its decomposition [25].

Table 3. The result of executing the proposed HA on IBM Q [24].

| X | Y | C_{out} | SUM | Success Probability |
|---|---|---------|-----|---------------------|
| 0 | 0 | 0       | 0   | 0.8829              |
| 0 | 1 | 0       | 1   | 0.8001              |
| 1 | 0 | 0       | 1   | 0.7969              |
| 1 | 1 | 1       | 0   | 0.8239              |

Average success probability 0.8259

Table 4. The result of executing the proposed HS on IBM Q [24].

| X | Y | B_{out} | Diff | Success Probability |
|---|---|---------|------|---------------------|
| 0 | 0 | 0       | 0    | 0.7457              |
| 0 | 1 | 1       | 1    | 0.7387              |
| 1 | 0 | 0       | 1    | 0.7545              |
| 1 | 1 | 0       | 0    | 0.7436              |

Average success probability 0.7456

Table 5 shows the truth table of the addition operation for the proposed FA. In addition, it illustrates the success probability of finding the desired output after running the proposed FA on IBM Q [24], with a maximum success probability 0.7703 and an average success probability 0.6935. Figure 10b illustrates the proposed FA circuit implementation using Qiskit Python SDK [25].
Table 5. The result of executing the proposed FA on IBM Q [24].

| X  | Y  | C_{in} | C_{out} | SUM  | Success Probability |
|----|----|--------|---------|------|--------------------|
| 0  | 0  | 0      | 0       | 0    | 0.7088             |
| 0  | 0  | 1      | 0       | 1    | 0.7117             |
| 0  | 1  | 0      | 0       | 1    | 0.7703             |
| 0  | 1  | 1      | 1       | 0    | 0.7620             |
| 1  | 0  | 0      | 0       | 1    | 0.6081             |
| 1  | 0  | 1      | 1       | 0    | 0.6854             |
| 1  | 1  | 0      | 1       | 0    | 0.6879             |
| 1  | 1  | 1      | 1       | 1    | 0.6138             |

Average success probability 0.6935

The proposed FS is tested on IBM Q [24]. For each possible input, the desired output is produced with certain success probability, as depicted in Table 6. The proposed FS circuit implementation using Qiskit Python SDK [25] is depicted in Figure 10c, with a maximum success probability 0.76 and an average success probability 0.6829.

Table 6. The results of executing the proposed FS on IBM Q [24].

| X  | Y  | B_{in}  | B_{out} | Diff | Success Probability |
|----|----|---------|---------|------|--------------------|
| 0  | 0  | 0       | 0       | 0    | 0.76               |
| 0  | 0  | 1       | 1       | 1    | 0.6376             |
| 0  | 1  | 0       | 1       | 1    | 0.6531             |
| 0  | 1  | 1       | 1       | 0    | 0.6632             |
| 1  | 0  | 0       | 0       | 1    | 0.7121             |
| 1  | 0  | 1       | 0       | 0    | 0.7266             |
| 1  | 1  | 0       | 0       | 0    | 0.7033             |
| 1  | 1  | 1       | 1       | 1    | 0.6075             |

Average success probability 0.6829

Table 7 studies the proposed FA and the proposed FS, and compares them with relevant work [18–20,35–40]. It shows that the proposed FA has no garbage output and QC of six, which is the best among the designs in the comparison. In addition, the proposed FA design has CI of one, which is similar to [20,35,37,39], and GC of two similar to [18,20,35]. The delay of the proposed FA design has the smallest delay among the relevant work that considered circuit delay as a metric. Overall, the proposed FA design functions with better results when it is compared to relevant work.

In addition, Table 7 describes the proposed FS design in terms of known metrics and compares it to relevant work [17–20,36,37,40]. Table 7 shows that the proposed FS design has no GO and no CI which is the best results obtained when comparing it to relevant work. Additionally, the proposed FS design shows QC of eight and delay of seven which are high when it is compared to the FS design in [17], but the FS design in [17] did not consider other metrics such as GO, CI or GC. Overall, the proposed FS design shows better results when it is compared to relevant designs.
Table 7. Comparing the proposed FA and FS designs with relevant work, where (−) refers to metric not calculated by the others.

| Reference | Function | QC | GO | CI | GC | Delay |
|-----------|----------|----|----|----|----|-------|
| Proposed design Section 3.2 | FA | 6 | 0 | 1 | 2 | 4Δ |
| Proposed design Section 3.3 | FS | 8 | 0 | 0 | 4 | 7Δ |
| [35] | FA | − | 2 | 1 | 2 | − |
| [36] | FA/FS | 24 | 6 | 7 | 6 | 16Δ |
| Design 1 [37] | FA/FS | 21 | 3 | 3 | 8 | − |
| Design 2 [37] | FA/FS | 14 | 3 | 1 | 4 | − |
| Design 3 [37] | FA/FS | 10 | 3 | 1 | 4 | − |
| [38] | FA | − | 3 | 2 | 4 | − |
| [39] | FA | 10 | 2 | 1 | 3 | − |
| [40] | FA | 6 | − | − | − | 6Δ |
| [41] | FA/FS | 19 | − | − | 8 | − |
| [42] | FA/FS | 11 | 2 | 0 | 1 | − |
| [43] | FA/FS | 8 | 2 | 1 | 2 | 8Δ |

5. Conclusions

In this paper, we proposed an efficient designs of reversible half and full addition and subtraction circuits. The addition and subtraction operations are the basic operations in the ALUs, since the multiplication and division operations can be obtained using repeated addition and subtraction. The proposed designs are based on a universal single type library G which can be used to construct cheap reversible circuits using similar building blocks. The proposed full adder and full subtractor are built using two G gates, which works as a 1-bit ALU and can also be extended to work on any number of bits using multiplexers. The presented designs are multi-function circuits which are capable of performing additional logical operations: AND, OR, NOT, XOR, XNOR, NOR and NAND. These logical gates are sufficient to construct classical universal libraries, e.g., {AND, OR, NOT}. The proposed designs show reduction in quantum cost, gate count, constant input, and delay with zero garbage output, when they are compared to relevant work in the literature.

The experimental results show the efficiency of the proposed designs when they are implemented and tested on IBM Q to measure the success probability of obtaining the correct results. It is shown that the average success probability of the half adder is 0.8259 with a maximum success probability 0.8239. The average success probability of the half subtractor is 0.7456 with a maximum success probability 0.7545. In addition, the average success probability for the full adder is 0.6935, with a maximum success probability 0.7703. Finally, the full subtractor illustrates 0.6829 average success probability with a maximum success probability of 0.76. An effort is required to improve the average success probability of the proposed designs on different quantum computer architectures. Further effort is required to expand the proposed design to its elemental gates and provide optimization techniques to reduce the gate count and improve the efficiency of the designs.

Author Contributions: Conceptualization, M.O. and K.E.-W.; supervision, M.O.; software M.O. and K.E.-W.; validation M.O. and K.E.-W.; writing the paper, M.O. and K.E.-W. Both authors have read and agreed to the published version of the manuscript.

Funding: This project was supported financially by the Academy of Scientific Research and Technology (ASRT), Egypt, Grant No. 6614, (ASRT) is the 2nd affiliation of this research.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.
Acknowledgments: We acknowledge the use of IBM Quantum services for this paper. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Bennett, C.H. Logical reversibility of computation. *IBM J. Res. Dev.* **1973**, *17*, 525–532. [CrossRef]

2. Fredkin, E.; Toffoli, T. Conservative logic. *Int. J. Theor. Phys.* **1982**, *21*, 219–253. [CrossRef]

3. Gariepelly, R.; Kiran, P.M.; Kumar, A.S. A review on reversible logic gates and their implementation. *Int. J. Emerg. Technol. Adv. Eng.* **2013**, *3*, 417–423.

4. De Vos, A.; Desoete, B.; Jania, F.; Nogawski, A. Control Gates as Building Blocks for Reversible Computers; Proceedings Patmos 2001 Conference; Springer: New York, NY, USA, 2001.

5. De Vos, A.; Desoete, B.; Adamski, A.; Pietrzak, P.; Śliwiński, M.; Widerski, T. Design of reversible logic circuits by means of control gates. In *International Workshop on Power and Timing Modeling, Optimization and Simulation*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 255–264.

6. Orlov, A.; Amlani, I.; Bernstein, G.; Lent, C.; Snider, G. Realization of a functional cell for quantum-dot cellular automata. *Science* **1997**, *277*, 928–930. [CrossRef]

7. Amlani, I.; Orlov, A.O.; Toth, G.; Bernstein, G.H.; Lent, C.S.; Snider, G.L. Digital logic gate using quantum-dot cellular automata. *Science* **1999**, *284*, 289–291. [CrossRef] [PubMed]

8. Feynman, R.P. Quantum mechanical computers. *Found. Phys.* **1986**, *16*, 507–531. [CrossRef]

9. Deutsch, D. Quantum theory, the Church–Turing principle and the universal quantum computer. *Proc. R. Soc. London. A Math. Phys. Sci.* **1985**, *400*, 97–117.

10. Lloyd, S. A potentially realizable quantum computer. *Science* **1993**, *261*, 1569–1571. [CrossRef] [PubMed]

11. Buhrman, H.; Cleve, R.; Wigderson, A. Quantum vs. classical communication and computation. In Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, Dallas, TX, USA, 24–26 May 1998; pp. 63–68.

12. Grover, L.K. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.* **1997**, *79*, 325. [CrossRef]

13. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **1999**, *41*, 303–332. [CrossRef]

14. Nielsen, M.A.; Chuang, I. Quantum Computation and Quantum Information; Cambridge University Press: Cambridge, UK, 2002.

15. Mohammadi, M.; Eshghi, M. On figures of merit in reversible and quantum logic designs. *Quantum Inf. Process.* **2009**, *8*, 297–318. [CrossRef]

16. Patterson, D.A.; Hennessy, J.L. *Computer Organization and Design ARM Edition: The Hardware Software Interface*; Morgan Kaufmann: San Francisco, CA, USA, 2016.

17. Thapliyal, H.; Ranganathan, N. A new design of the reversible subtractor circuit. In Proceedings of the 2011 11th IEEE International Conference on Nanotechnology, Portland, OR, USA, 15–18 August 2011; pp. 1430–1435.

18. Gupta, A.; Singla, P.; Gupta, J.; Maheshwari, N. An improved structure of reversible adder and subtractor. *arXiv* 2013, arXiv:1306.1889.

19. Moghimi, S.; Reshadinezhad, M.R. A Novel 4×4 Universal Reversible Gate as a Cost Efficient Full Adder/Subtractor in Terms of Reversible and Quantum Metrics. *Int. J. Mod. Educ. Comput. Sci.* **2015**, *7*, doi:10.5815/ijmecs.2015.11.04. [CrossRef]

20. Montaser, R.; Younes, A.; Abdel-Aty, M. New design of reversible full adder/subtractor using R gate. *Int. J. Theor. Phys.* **2019**, *58*, 167–183. [CrossRef]

21. Younes, A. On the universality of n-bit reversible gate libraries. *Appl. Math. Inf. Sci.* **2015**, *9*, 2579.

22. Montaser, R.; Younes, A.; Abdel-Aty, M. New Design of Universal Reversible Gate Library. *Quantum Matter* **2017**, *6*, 89–96. [CrossRef]

23. Osman, M.; Younes, A.; Ismail, G.; Farouk, R. An Improved Design of n-Bit Universal Reversible Gate Library. *Int. J. Theor. Phys.* **2019**, *58*, 2531–2549. [CrossRef]

24. Team I.Q. IBM Q 5 Santiago Backend Specification V1.0.0; IBM: Armonk, NY, USA, 2018.

25. Qiskit: An Open-source Framework for Quantum Computing. 2019. Available online: https://qiskit.org/ (accessed on 8 August 2021).

26. Mano, M.M.; Ciletti, M. *Digital Design: With an Introduction to the Verilog HDL*; Pearson: New York, NY, USA, 2013.

27. Katz, R.H.; Borriello, G. Contemporary Logic Design. 2005. Available online: https://www.amazon.com/Contemporary-Logic-Design-Randy-Katz/dp/0201308576 (accessed on 8 August 2021).

28. Yang, G.; Song, X.; Perkowski, M.A.; Hung, W.N.; Seo, C.J. Minimal universal library for n× n reversible circuits. *Comput. Math. Appl.* **2008**, *56*, 160–165. [CrossRef]

29. Toffoli, T. Reversible computing. In *International Colloquium on Automata, Languages, and Programming*; Springer: Berlin/Heidelberg, Germany, 1980; pp. 632–644.

30. Montaser, R.; Younes, A.; Abdel-Aty, M. Improving the quantum cost of NCT-based reversible circuit. *Quantum Inf. Process.* **2015**, *14*, 1249–1263. [CrossRef]
31. Ali, M.B.; Hirayama, T.; Yamanaka, K.; Nishitani, Y. Quantum cost reduction of reversible circuits using new Toffoli decomposition techniques. In Proceedings of the 2015 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 7–9 December 2015; pp. 59–64.
32. Shende, V.V.; Prasad, A.K.; Markov, I.L.; Hayes, J.P. Synthesis of reversible logic circuits. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* 2003, 22, 710–722. [CrossRef]
33. Maslov, D.; Miller, D.M. Comparison of the cost metrics for reversible and quantum logic synthesis. *arXiv* 2005, arXiv:quant-ph/0511008.
34. Yang, G.; Song, X.; Hung, W.N.; Perkowski, M.A.; Seo, C.J. Synthesis of reversible circuits with minimal costs. *Calcolo* 2008, 45, 193–206. [CrossRef]
35. Ni, L.; Guan, Z.; Zhu, W. A general method of constructing the reversible full-adder. In Proceedings of the 2010 Third International Symposium on Intelligent Information Technology and Security Informatics, Jian, China, 2–4 April 2010; pp. 109–113.
36. Morrison, M.; Ranganathan, N. Design of a reversible ALU based on novel programmable reversible logic gate structures. In Proceedings of the 2011 IEEE Computer Society Annual Symposium on VLSI, Chennai, India, 4–6 July 2011; pp. 126–131.
37. Rangaraju, H.; Venugopal, U.; Muralidhara, K.; Raja, K. Design of efficient reversible parallel Binary adder/subtractor. In *International Conference on Advances in Communication, Network, and Computing*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 83–87.
38. Bruce, J.; Thornton, M.A.; Shivakumaraiah, L.; Kokate, P.; Li, X. Efficient adder circuits based on a conservative reversible logic gate. In Proceedings of the IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design (ISVLSI 2002), Pittsburgh, PA, USA, 25–26 April 2002; pp. 83–88.
39. Babu, H.M.H.; Islam, M.R.; Chowdhury, S.M.A.; Chowdhury, A.R. Synthesis of full-adder circuit using reversible logic. In Proceedings of the 17th International Conference on VLSI Design. Proceedings, Mumbai, India, 9 January 2004; pp. 757–760.
40. Thapliyal, H.; Srinivas, M. A new reversible TSG gate and its application for designing efficient adder circuits. *arXiv* 2006, arXiv:cs/0603091.