A Space-Efficient Design for a Reversible Floating Point Adder in Quantum Computing

Trung Duc Nguyen, Keio University
Rodney Van Meter, Keio University

Reversible logic has applications in low-power computing and quantum computing. However, there are few existing designs for reversible floating-point adders and none suitable for quantum computation. In this paper we propose a space-efficient reversible floating-point adder, suitable for binary quantum computation, improving the design of Nachtigal et al. [Nachtigal et al. 2011]. Our work focuses on improving the reversible designs of the alignment unit and the normalization unit, which are the most expensive parts. By changing a few elements of the existing algorithm, including the circuit designs of the RLZC (reversible leading zero counter) and converter, we have reduced the cost about 68%. We also propose fault-tolerant quantum designs. The KQ for our fault-tolerant design is almost sixty times as expensive as for a 32-bit fixed-point addition. We note that the floating-point representation makes in-place, truly reversible arithmetic impossible, requiring us to retain both inputs, which limits the sustainability of its use for quantum computation.

Categories and Subject Descriptors: C.1.m [**Processor Architectures**]: Miscellaneous; B.2.0 [**Arithmetic and Logic Structures**]: General; B.m [**Hardware**]: Miscellaneous

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: reversible circuit, quantum computing, low-power computing, nano technology, IEEE-754 specification, floating-point arithmetic

ACM Reference Format:
Gang Zhou, Yafeng Wu, Ting Yan, Tian He, Chengdu Huang, John A. Stankovic, and Tarek F. Abdelzaher, 2010. A multifrequency MAC specially designed for wireless sensor network applications. *ACM Trans. Embedd. Comput. Syst.* 9, 4, Article 39 (March 2010), 18 pages.
DOI : http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

In irreversible systems, erasure of a single bit generates $kT \ln 2$ joules of heat energy where k is Boltzmann's constant of $1.38 \times 10^{-23} m^2 s^{-2} kg K^{-1}$ and T is the absolute temperature of the environment. Based on this observation, Landauer showed that for a reversible computer the energy dissipation is exactly $kT \ln 1$ which is equal to zero [Landauer 1961]. This means reversible logic has applications in low power computing. Additionally, quantum computing inherently uses reversible computing because operations other than measurement in quantum computing are unitary and work as reversible functions. For these reasons, reversible circuit design is receiving a lot of attention from quantum researchers.

Some quantum algorithms would benefit from the availability of a library of floating point operations. Algorithms that focus on physical phenomena, such as quantum chemistry [Brown et al.

An abstract of this paper was presented at AQIS 2013, Chennai, India, August 2013.
A preprint of the full paper is posted at http://arxiv.org/abs/1306.3760.
This research is supported by the Cabinet Office, Government of Japan and the Japan Society for the Promotion of Science (JSPS) through the Funding Program for World-Leading Innovative R&D on Science and Technology (FIRST Program).
Author's addresses: Trung Duc Nguyen and Rodney Van Meter, Faculty of Environment and Information Studies, Keio University, 5322 Endo, Fujisawa, Kanagawa, Japan.

DOI : http://dx.doi.org/10.1145/0000000.0000000

2010; Kassal et al. 2011] and quantum field theory calculations [Jordan et al. 2012], seem especially likely to be able to take advantage of this capability. The quantum field theory algorithm of Jordan, Lee and Preskill, for example, discretizes the field strength, representing it in multiples of a base value δ_ϕ and setting a maximum magnitude $\phi_{\max} = 2^n \delta_\phi$ by using an n-bit variable, in standard fixed-point practice. The availability of a practical floating point representation would allow accurate representation of a far broader dynamic range of field magnitude. Implementations of Harrow, Hassidim and Lloyd's algorithm for linear systems [Clader et al. 2013; Harrow et al. 2009] and Jozsa's variant of Hallgren's algorithm that uses real numbers may also benefit [Jozsa 2003]. The new quantum machine learning algorithm of Lloyd, Mohseni, and Rebentrost uses real-valued vector elements (stored in a quantum RAM, or QRAM [Giovannetti et al. 2008a; 2008b]) and calculates the distance between vectors, operations normally assigned to floating point representations in classical computers [Lloyd et al. 2013].

While many reversible fixed point adder designs for quantum computation have been introduced [Beckman et al. 1996; Choi and Van Meter 2012; Cuccaro et al. 2004; Draper et al. 2006; Takahashi 2009; Takahashi and Kunihiro 2005; Van Meter and Itoh 2005; Vedral et al. 1996], we are aware of only one design for a floating-point adder, by Nachtigal, Thapliyal and Ranganathan (NTR), and this design is expensive. Our proposed design solves this problem by improving the expensive parts in the NTR design [Nachtigal et al. 2011]. About 68% of the cost has been eliminated. Moreover, the NTR design as presented leaves many temporary variables in a dirty state, making it unsuitable as-is for quantum computing; our design reduces this number and shows how to compose this design in a fully-reversible setup.

A truly reversible circuit generally calculates $\langle A, B\rangle \xrightarrow{U} \langle A, f(A, B)\rangle$ where each element of the tuple is a fixed-size register and U is a unitary operation or set of operations that realizes $f(A, B)$. The NTR circuit actually calculates $\langle A, B, 0, 0\rangle \xrightarrow{U} \langle A, B, A + B, G\rangle$ where A, B and $A + B$ are single precision floating point numbers and G is a large amount of ancillary data left in a garbage state. We adapt Bennett's original reversible formulation,

$$\begin{aligned}\langle A, B, 0, 0, 0\rangle &\xrightarrow{U}\langle A', B', f(A, B), G, 0\rangle \\ &\xrightarrow{CNOT}\langle A', B', f(A, B), G, f(A, B)\rangle \\ &\xrightarrow{U^\dagger}\langle A, B, 0, 0, f(A, B)\rangle.\end{aligned} \tag{1}$$

This reduces the garbage output, but cannot solve the fundamental problem that floating point addition is not 1:1, requiring us to retain both inputs as well as the output. Thus, quantum circuits that require many floating point operations may result in unsustainable growth of memory resources.

This paper is divided into six parts. Section 2 reviews reversible logic, evaluation metrics and the IEEE-754 single-precision floating point specification. Section 3 briefly describes the floating-point adder algorithm while Section 4 shows our proposed designs. The comparison between the NTR design and our proposed design and fault-tolerant designs are in Section 5 and 6 respectively. Section 7 concludes.

2. BACKGROUND

2.1. Metrics for Evaluating Quantum Circuits

In this paper, we will evaluate circuits using circuit depth (execution time) and Steane's KQ metric (total resource cost), as well as adopting Nachtigal's approach of using the quantum cost, the number of constant inputs and the number of garbage outputs. We define the quantum cost as the number of basic gates, while garbage output is the number of unnecessary output qubits which must be cleaned up later. These will be denoted as G in the output of our circuits. Constant inputs are qubits with a fixed value, often used to emulate Boolean logic in reversible logic, or as constant values in algorithms. Reversible circuits must always have the same number of inputs and outputs, thus

the number of constant plus variable inputs must be equal to the number of useful variable outputs plus garbage. Fig. 1(a) shows the unitary operator of the TR gate [Thapliyal and Ranganathan 2009] which we use rather than the Peres gate [Peres 1985] that Nachtigal favors, while Fig. 1(b) shows the quantum Barenco decomposition [Barenco et al. 1995]. This gate has quantum cost of 4, no constant inputs, and 2 garbage outputs if we only use the third output. The Peres gate's unitary matrix and Barenco decomposition are shown in Fig. 2. The V and $V^\dagger$ operators, which are unique to quantum computation, behave as follows:

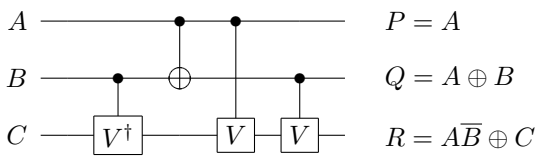
$$VV = V^\dagger V^\dagger = X,$$
$$VV^\dagger = V^\dagger V = I,$$

where X is the Pauli gate corresponding to classical NOT. The V operator is

$$V = \frac{1}{2}\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix}.$$

$$\begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0 \end{pmatrix}$$

(a) Unitary matrix

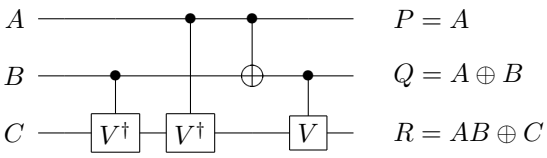


(b) Barenco decomposition

Fig. 1: TR gate

$$\begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \end{pmatrix}$$

(a) Unitary matrix



(b) Barenco decomposition

Fig. 2: Peres gate

Quantum computers are far more susceptible to making errors than conventional digital computers, and some method of controlling and correcting those errors will be needed to prevent a quantum computer from making undetected errors in the computation. A device that works effectively even when its elementary components are imperfect is said to be fault-tolerant.

In fault-tolerant quantum computation [Preskill 1997], we use a group of physical qubits to represent a smaller number of logical qubits, adding redundant information to protect against errors [Devitt et al. 2013; Raussendorf 2012; Terhal 2013]. The encoding of logical information in a quantum error correcting code has the consequence of making logical gates harder to execute. The most commonly used set of gates sufficient for universal fault-tolerant quantum computation is the Clifford+T

set [Matsumoto and Amano 2008]. Because of this we also use another metric to evaluate a circuit design. The number of T or $T^\dagger$ gate is used to calculate quantum cost and T-depth, which is the number of steps using a T or $T^\dagger$ gate. The reason is because T or $T^\dagger$ gates are the most expensive elements in Clifford+T set to implement fault-tolerantly. The T gate modifies the phase of the quantum state without changing the probability of measuring a $|0\rangle$ or $|1\rangle$. The T gate is

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix}.$$

Fig. 3 shows the decomposition of the fault-tolerant Fredkin gate [Amy et al. 2013]. In this circuit, the T-depth is 4.

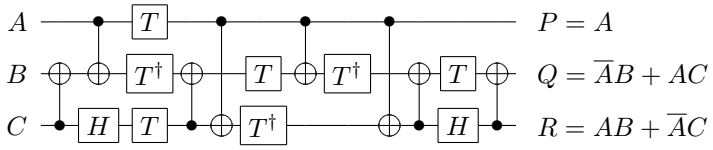


Fig. 3: Decomposition of Fault-Tolerant Fredkin gate.

2.2. IEEE-754 Floating-Point

The IEEE-754 single-precision floating-point format called binary32 [Wikipedia] uses 32 bits (a 1-bit sign, an 8-bit exponent and a 23-bit mantissa) to represent a floating point number. The sign bit determines the sign of the number, while the exponent is an 8 bit unsigned integer from 0 to 255. The true significand, called the mantissa, includes 23 fraction bits to the right of the binary point and an implicit leading bit (to the left of the binary point) with value 1 unless the exponent is all zeros. Thus only 23 fraction bits of the significand appear in the memory format but the total precision is 24 bits. Suppose that we have a floating-point number with sign bit s, exponent e and mantissa $m_{22}m_{21}..m_0$, then the value of the floating point number is calculated as follows:
$(-1)^s \times (1.m_{22}m_{21}...m_0) \times 2^{e-127}$

This format also requires 3 extra bits during computation known as the guard bit, round bit and sticky bit, which must be preserved during the right shifts. The guard and round bits are just two extra bits of precision that are used in calculations. The sticky bit is an indication of what is or could be in less significant bits that are not kept. If a value of 1 ever is shifted into the sticky bit position, that sticky bit remains a 1 (”sticks” at 1), despite further shifts.

A natural consequence of the limited precision of floating point is the rounding of results. And example using 4 decimal digits would be $1.000 + 0.0001 = 1.000$. Adding any value from 0.0000 to 0.0004, or possibly higher depending on rounding rules, would give the same result. Thus, the addition operation is not 1:1.

3. FLOATING-POINT ADDER OVERVIEW

In this section the basics of a floating-point adder algorithm will be briefly summarized with attention to the demands of reversibility. Two 32-bit IEEE-754 single-precision floating-point num bers A and B are to be added. Before two numbers can be added, they must be aligned. If the exponents are not equal, the smaller number's exponent is incremented until its exponent reaches the larger number's, in conjunction with shifting the smaller number's mantissa to the right. Once the exponents are equal, the mantissas can be summed. The sum is normalized and rounded at the end. Fig. 4 shows the general algorithm adapted to show constant inputs and garbage outputs. The garbage outputs are eventually cleaned by reversing this circuit using Bennett's method.

Reversible Conditional Swap. A reversible conditional swap is necessary because we need to determine which number has the smaller exponent and then input it to the reversible alignment step. If expA $<$ expB (where expA stands for the exponent of A) then swap the two numbers, otherwise

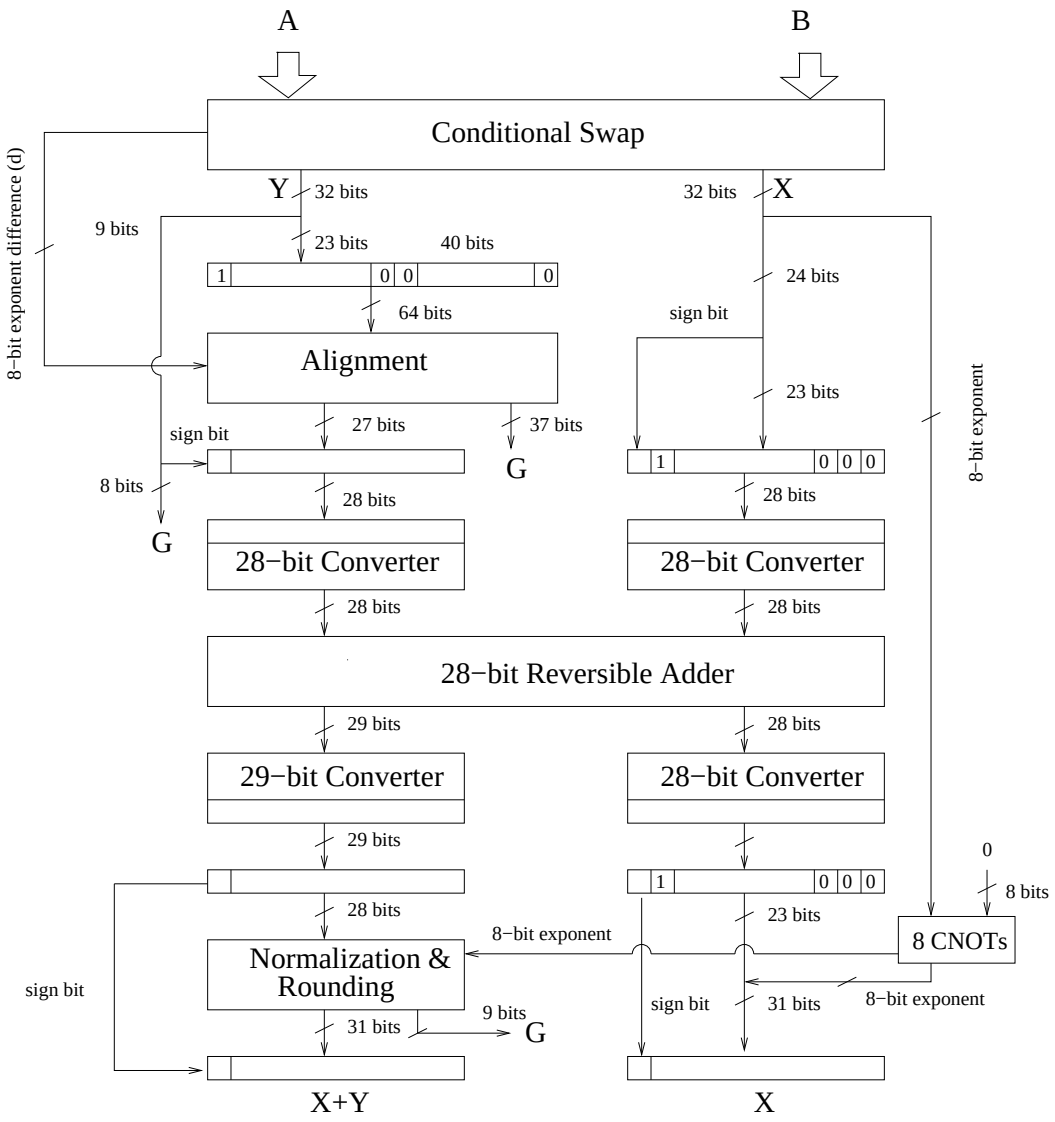


Fig. 4: Overview of the algorithm for a Floating Point Adder.

do nothing. After this step, the number with the smaller exponent always comes out in the Y output, which connects to the reversible barrel shifter [Gorgin and Kaivani 2007] in the next step.

Reversible Alignment. We need reversible alignment because we can only add two mantissas when the two exponents are equal, so we need a reversible right shifter to shift the mantissa of the number with smaller exponent, in conjunction with increasing the smaller exponent. The shifted amount is the difference between the two exponents. Because the IEEE-754 floating-point single-precision specification uses an 8-bit exponent, the difference of the exponents is up to 256.

The IEEE-754 specification also requires 3 extra bits as described above. Thus, we need a sticky bit cascade unit to calculate the sticky bit after shifting. The sticky bit is calculated ORing together the 27^{th} to 256^{th} bits.

Two's complement Conversion. The IEEE-754 specification represents numbers in sign-magnitude format (1 sign bit, 23 mantissa bits). To add two mantissas after the alignment step, the two numbers will be represented in two's complement format. After the addition, the result will be converted back to sign-magnitude format. Thus, we need *sign-magnitude to two's complement* reversible converters and *two's complement to sign-magnitude* reversible converters before and after the addition.

Reversible Addition. After the *sign-magnitude to two's complement* conversion, the addition is done by a reversible adder which is constructed from 27 RFA (Reversible Full Adder) gates and one RHA (Reversible Half Adder) gate.

Reversible Normalization and Rounding. After the addition, the result may have a number of leading zero bits or have one more bit with value of one at the most significant bit (MSB). The normalization is needed to adjust the result so that it conforms to the floating-point number format. In normalization, if a shift is required, it is either a one place right shift or a multiple-place left shift. If the MSB has a value of one, one place of right shift takes place and the 8-bit exponent is passed through a reversible conditional increment unit. Otherwise, one or several places of left shift is needed in conjunction with a corresponding decrement of the 8-bit exponent.

4. DETAILED DESIGN

In the NTR design, two parts are the main causes of the large quantum cost, the reversible alignment unit and the reversible normalization unit, with 12,312 and 2,009 gates respectively. Our proposed design focuses mainly on these parts, and improves some other parts in smaller ways.

4.1. Reversible Conditional Swap

In our proposed design, the actual swap is done by a bank of 32 Fredkin gates but we use 7 RFS (reversible full subtractor) and one RHS (reversible half subtractor) to construct the reversible subtractor as shown in Fig. 5. In the existing design of RHS and RFS [Thapliyal and Ranganathan 2011] in Fig. 6 and Fig. 8, the quantum costs are 4 and 6 respectively. To reduce the garbage output and reuse them we proposed the new design in Fig. 7 and Fig. 9. In Fig. 5, the A_i^e and B_i^e denote the exponent bits of the two floating-point numbers, while A_i^m and B_i^m are the mantissa bits and A^s and B^s are sign bits. The borrow bits b_i from previous RHS or RFS are passed to the next RFS. The high order output bit b_7 will be used as a conditional signal for swap and the difference which is output in two's complement format $b_7 d_7..d_0$ will be used for the alignment unit as the shifted amount after conversion to sign-magnitude format.

4.2. Reversible Alignment

Although the difference in the exponents may be up to 256, we observe that the mantissa with guard bit and round bit is only 26 bits. Thus, when the difference is greater than 26, the smaller number is effectively discarded and the result is simply the larger number. Thus, we propose using a shift limiter unit, which outputs the smaller of the exponent difference and 26. This output is used to control the barrel shifter. We only need a reversible barrel shifter for 50 input bits and 23 reversible OR gates for the reversible sticky bit cascade unit. We choose to use a (64, 6) reversible barrel shifter [Hashmi and Babu 2010].

One difficulty is the floating point sticky bit, which is the OR of all of the bits shifted past it. Calculating an OR function in reversible logic is difficult, requiring us to keep additional ancillae. In our proposed design, we use TR gates as in Fig. 1, first applying a NOT gate to the A input and

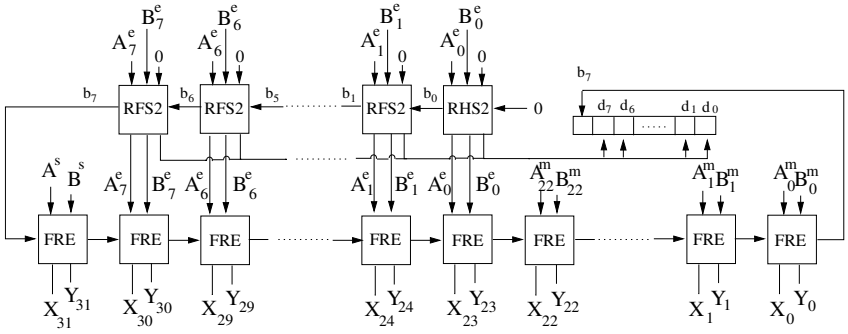
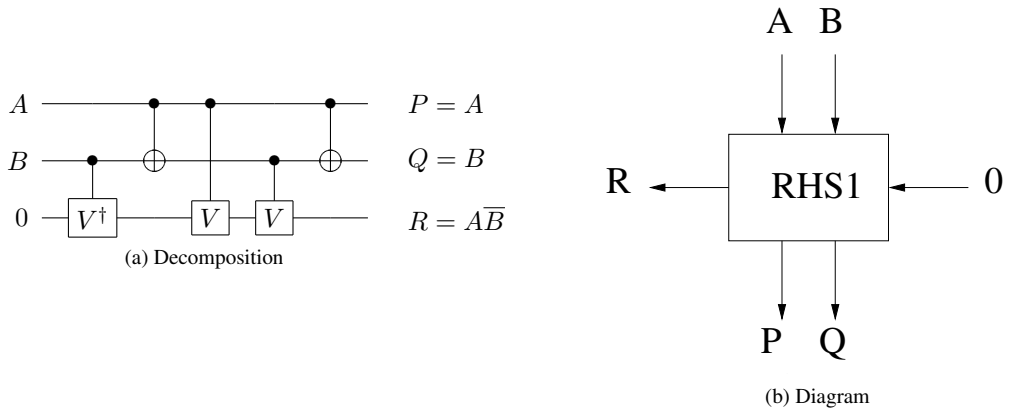


Fig. 5: Proposed Conditional Swap. The output register d is the difference of the exponents, which later will be fed into the shift limiter.



(a) Decomposition

(b) Diagram

Fig. 6: Diagram and decomposition of the existing Reversible Half Subtractor (RHS1).

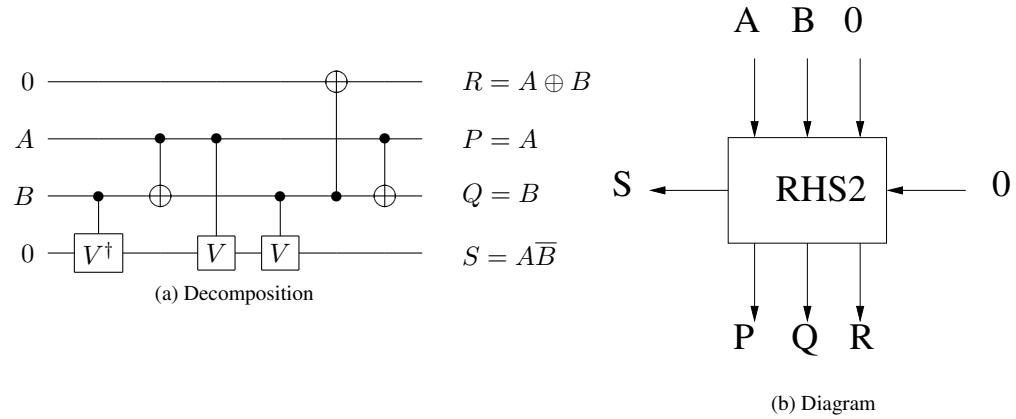
setting the C input to the constant value 1. The OR operation between A and B input comes out in the R output.

Fig. 10 shows the proposed design for the alignment unit at the block level.

The shift limiter works in the same way as conditional swap at the first step except that only 8 bits are swapped instead of 32 bits. It is also slightly different in that we only need the smaller of the number 26 and the exponent's difference so we don't need to use the design of RHS2 in Fig. 7, which is used to reduce the number of garbage outputs. Instead, we use the RHS1 in Fig. 6. The design of the shift limiter is shown in Fig. 11.

4.3. Reversible Converter

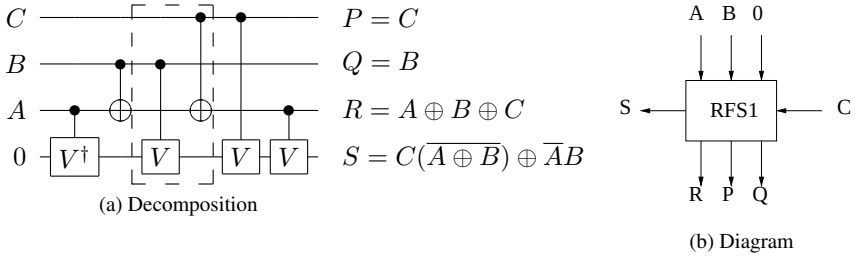
Suppose that we have a n-bit sign-magnitude number. The algorithm for converting a sign-magnitude number to two's complement is to invert all of the bits and add 1. To add 1 to the inverted bits we only need RHA gates instead of RFA gates combined with RHA gate, because we just need to add the carry bit of the previous operation. We realized that one TR gate can do both at the same time. In addition, we observe that after conversion the least significant bit (LSB) does not change and the carry bit for the next bit is the inverse of the LSB. Therefore, we only need 1 CNOT gate and $n - 2$ TR gates. Fig. 12 shows the new design and its diagram.



(a) Decomposition

(b) Diagram

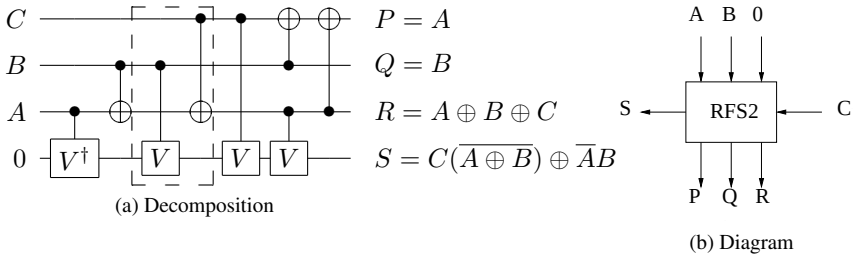
Fig. 7: Diagram and decomposition of our proposed Reversible Half Subtractor (RHS2).



(a) Decomposition

(b) Diagram

Fig. 8: Diagram and decomposition of the existing Reversible Full Subtractor (RFS1).



(a) Decomposition

(b) Diagram

Fig. 9: Diagram and decomposition of our proposed Reversible Full Subtractor (RFS2).

In the conversion step, we use a total four converters: two *28-bit sign-magnitude to two's complement* converters for pre-addition, one *28-bit two's complement to sign-magnitude* converter and one *29-bit two's complement to sign-magnitude* for post-addition. We use one *29-bit two's complement to sign-magnitude* to add one more high order bit to avoid overflow during the addition. See Fig. 4 for details.

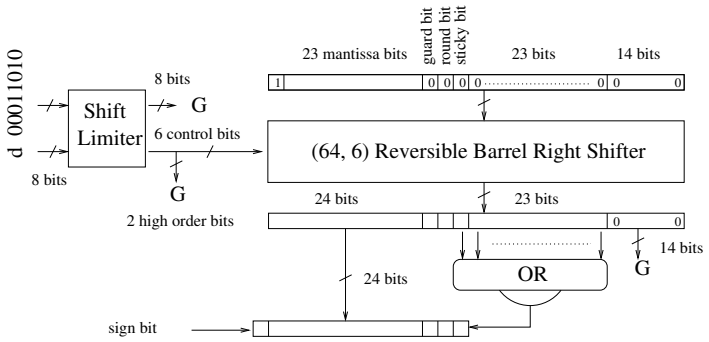


Fig. 10: Proposed design for Reversible Alignment Unit. The constant value in the top left input is the number 26, the upper limit for our shift distance. d is the exponent's difference.

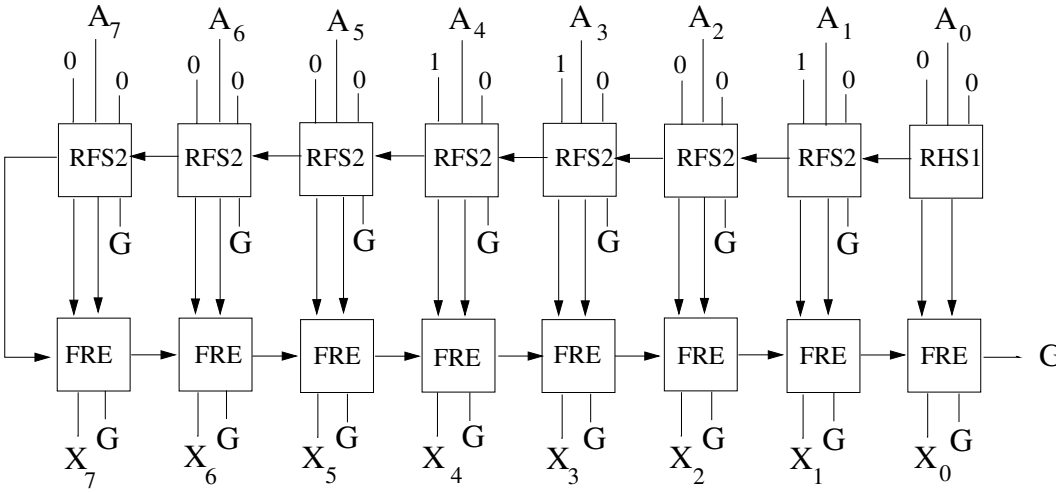


Fig. 11: The shift limiter works as a reversible conditional swap.

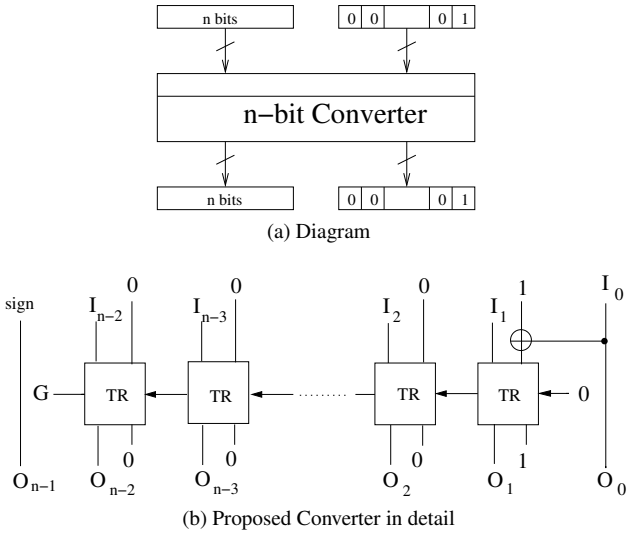
4.4. Reversible Normalization and Rounding

One of the problems is how to calculate the left shift amount (if needed). The left shift is the number of leading zero bits. The NTR design used a RLZCU (reversible leading zero counter unit), which requires $n - 1$ gates of RLZC (reversible leading zero counter) for n-bit input. However, the NTR design is not efficient and we propose a novel reversible leading zero counter design.

The output of reversible normalization is an array of 32 bits, but only the first 23 bits are needed. We used a *round toward zero* rounding algorithm. Thus, we just need to make the first bit and last bits become garbage outputs and no quantum gate is needed.

Our proposed design is constructed using one Toffoli gate, 3 TR gates, 2 NOT gates and 1 CNOT gate as shown in Fig. 13. Because each basic gate is counted as quantum cost 1, this design has quantum cost 20, garbage output 4 and constant input 4, respectively, and even reuses mantissa bits to reduce the number of garbage outputs.

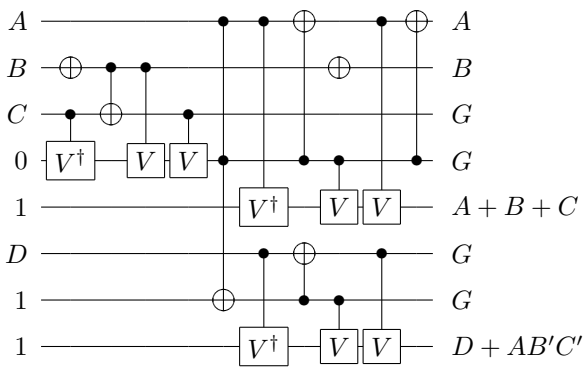
The shift is in conjunction with exponent addition or subtraction, thus, for these operations we use RHA gates, which are constructed from Peres gates. RFS gates are used to do the subtraction (if needed). We use the design in Fig. 8 for RFS gates and the design in Fig. 7 for RHS gate. The carry bits c_i and borrow bits b_i are passed to the next RHA or RFS gates as shown in Fig. 14. Because the



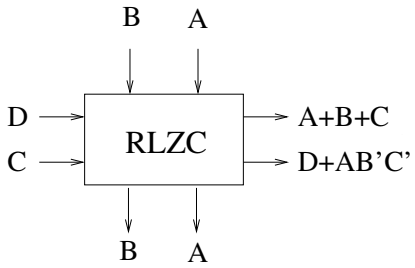
(a) Diagram

(b) Proposed Converter in detail

Fig. 12: Proposed design for Reversible n-bit Converter.



(a) Decomposition



(b) Diagram

Fig. 13: Diagram and our proposed design for RLZC.

RLZCU has a 5-bit output for 32-bit input while the input for RFS gates needs 8 bits, we add 3 zero

bits as the high order bits. Because the conditional right shifter is just a one place shift, we use the (28, 1) reversible barrel shifter described in [Gorgin and Kaivani 2007].

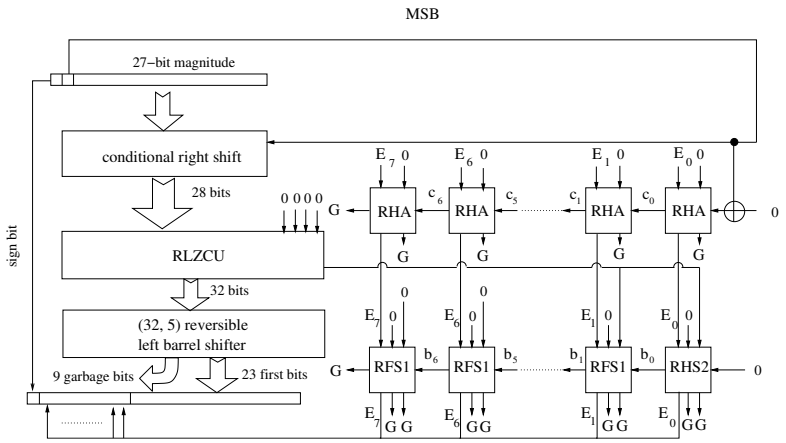


Fig. 14: Reversible Normalization.

Fig. 15 shows an example of using RLZCU for an 8-bit input. Suppose that the 8-bit input $X_7X_6...X_0$ is 00111010. The output 010 is produced on the bits $O_2O_1O_0$.

5. COMPARISON WITH THE NTR DESIGN

The NTR design is ambiguous in some places, and the evaluation apparently contains several errors. This section describes in details the difference between our proposed design and NTR design.

5.1. Reversible Conditional Swap

In the NTR design, the authors proposed using 9 HNG gates [Haghparast et al. 2008] as a reversible subtractor. The actual swap is done by a bank of 32 Fredkin gates [Fredkin and Toffoli 1982] controlled by the subtractor's high order output bit. While the NTR design includes fanout of exponent bits and produces garbage output, our design tries to reduce the quantum cost, garbage output and fanout by reusing the exponent bits after subtraction. In our proposed design, we use 7 RFS gates and 1 RHS gate to construct the reversible subtractor. The evaluation and comparison is shown in Table II.

5.2. Reversible Alignment

A (256, 8) reversible barrel shifter (256 input bits, 8 control lines) [Hashmi and Babu 2010] is used in the NTR design. This is one of the key reasons for the large cost of the NTR design. With our proposed design, we only use the (64, 6) reversible barrel shifter which reduces the cost significantly.

Nachtigal, Thapliyal and Ranganathan calculated a quantum cost of 4,632 for the alignment unit, using a Fredkin gate cost of 1 in their design. Assigning a cost of 5 to the Fredkin gate, we find a cost of 12,312 for this unit. According to [Hashmi and Babu 2010] the number of Fredkin gates and CNOT gates for a (256, 8) reversible barrel shifter should be 1,920 and 1,792 respectively. The authors apparently took the sum of these numbers, ignoring the Fredkin gate's non-unit cost, and arrived at a cost of 3,712, which we believe substantially understates the true cost of their design.

Using Peres gates to make a reversible OR gate, as in the NTR design, will have a quantum cost of at least 6, while our proposed design only requires 5.

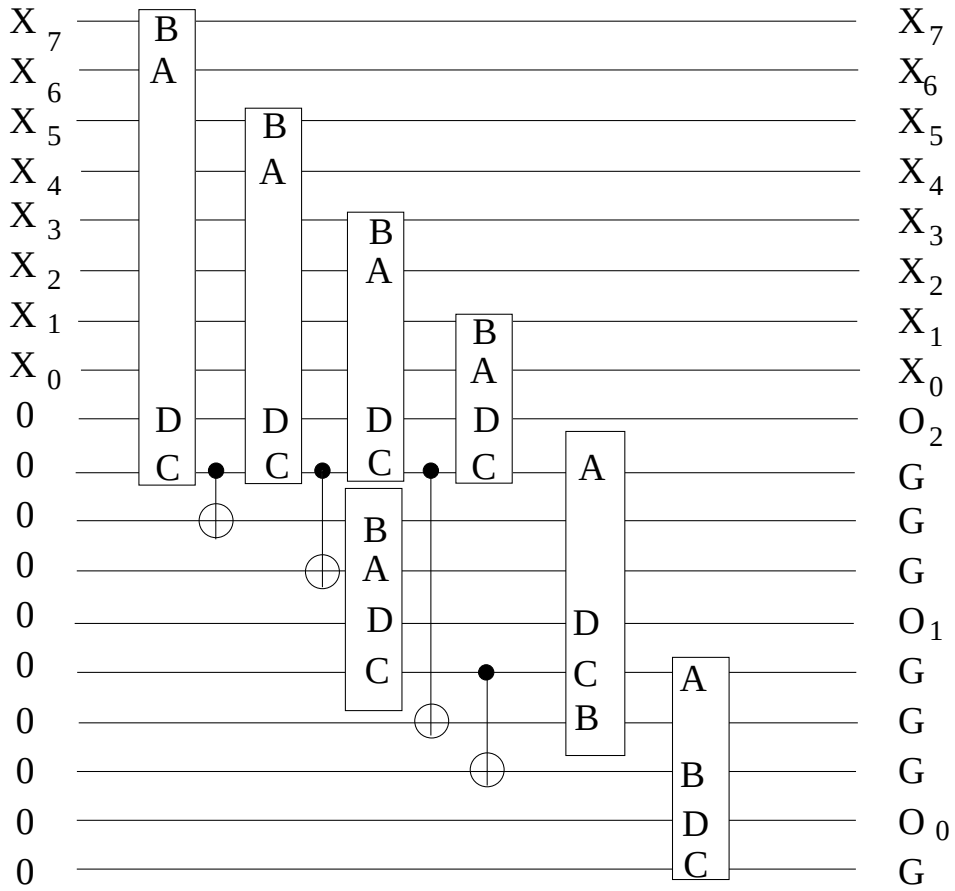


Fig. 15: An example of using RLZCU for 8 input bits.

	Quantum Cost	Garbage Ouput	Constant Input
NTR Design	$5n-4$	n	n
Proposed Design	$4n-7$	1	$n-1$

Table I: NTR Converter vs Proposed Converter.

5.3. Reversible Converter

The NTR design used $n-1$ Peres gates [Peres 1985] combined with $n-2$ NOT gates for both reversing bits and RHA gates, but we realized that one TR gate can do both at the same time. Our proposed design eliminates the NOT gates. We also save one more TR gate compare to the NTR design. The number of garbage ouput of our proposed design is reduced to 1 because most of the ancillae used in one converter can be reused in another conversion.

Table I shows a comparison between the NTR reversible converter and our proposed design for n-bit input. For the total evaluation of every conversion used in floating-point adder, see Table III. Note that our calculation for the conversions of our proposed design's cost differs from the authors of NTR design, who seem not to have included the cost of conversion for the exponent's difference after conditional swap. While the NTR design only uses 3 converters and leaves one operator of the addition in a dirty state, we use one more *two's complement to sign-magnitude* reversible converter to avoid that problem.

	Quantum Cost	Garbage Ouput	Constant Input
NTR Design	27	10	8
Proposed Design	20	4	4
Reduction Ratio	26%	60%	50%

Table II: NTR RLZC vs Proposed RLZC.

Stage	NTR Design			Proposed Design		
	QC	GO	CI	QC	GO	CI
Swap	238	19	27	220	0	9
Alignment	12312	2260	2022	2295	388	359
Addition	166	55	28	166	55	28
Conversion	454	94	94	450	56	55
Normalization	2009	498	484	1742	313	306
Rounding	0	9	0	0	9	0
Total	15179	2935	2655	4873	824	757

Table III: NTR Design vs Proposed Design.

5.4. Reversible Normalization and Rounding

By reducing the cost for each RLZC, our proposed design has substantially reduced the cost for the whole Reversible Leading Zero Counter Unit which is constructed from 31 RLZCs. Table II compares the NTR design and our proposed design for RLZC.

5.5. Overall Comparison

We have reduced the Quantum Cost by 68%, the Garbage Output by 72% and the Constant Input by 71.5%. Table III compares the NTR design and our proposed design in quantum cost, garbage output and constant input.

6. FAULT-TOLERANT DESIGN

In this paper, we have shown a lot of designs using controlled-V and controlled-$V^\dagger$ gates. But in quantum computing, the direct fault-tolerant implementation of controlled-V and controlled-$V^\dagger$ gates is very hard. To make our design implementable in quantum computing we need to use fault-tolerant forms of the TR gate, Peres gate, Fredkin gate and Toffoli gate. In this section we introduce these fault-tolerant circuit architectures.

Recently, M. Amy et al. [Amy et al. 2013] have introduced several depth-optimal decompositions of the Toffoli gate, Peres gate and TR gate. Fig. 16 shows the fault-tolerant designs of these gates.

The RLZC is mainly constructed from 3 TR gates and one Toffoli gate. We also propose a fault-tolerant design for RLZC gate by using the fault-tolerant designs of Toffoli gate and TR gate in Fig. 16. The design is shown in Fig.17 and this gate has T-depth of 11.

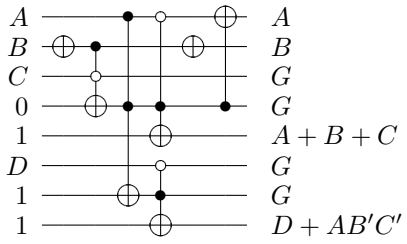
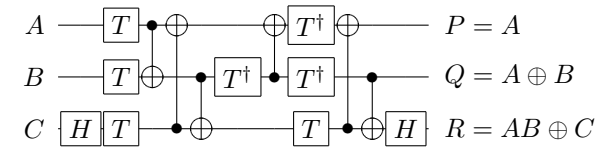
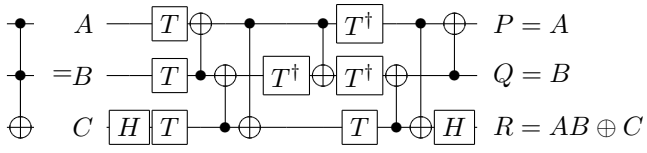


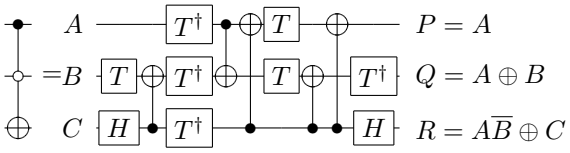
Fig. 17: Fault-Tolerant RLZC circuit



(a) Peres gate with T-depth 3



(b) Toffoli gate with T-depth 3

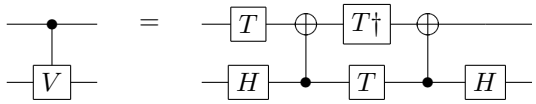


(c) TR gate with T-depth 4

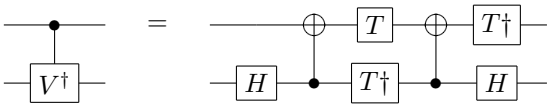
Fig. 16: Fault-tolerant architecture for Toffoli gate, Peres gate and TR gate

6.1. Controlled-V and Controlled-$V^\dagger$ Gates

To build the fault-tolerant designs for reversible half subtractor and reversible full subtractor described in previous sections, we incorporate fault-tolerant designs of the Controlled-V [Amy et al. 2013] and Controlled-$V^\dagger$ gates. The decomposition of the Controlled-$V^\dagger$ gate may be constructed from Controlled-V by putting adjoint operators of Controlled-V in reverse order. Fig. 18 shows the decomposition of these designs.



(a) Fault-Tolerant Controlled-V gate



(b) Fault-Tolerant Controlled-$V^\dagger$ gate

Fig. 18: Fault-Tolerant architecture for Controlled-V and Controlled-$V^\dagger$ gate

6.2. Reversible Full Adder

For our 28-bit adder, we use 27 RFA gates (Reversible Full Adder) and a one RHA gate, which is simply implemented by a single Peres gate. Fig. 19 shows the decomposition of RFA gate described in [Amy et al. 2013]. This RFA gate has T-depth of 2.

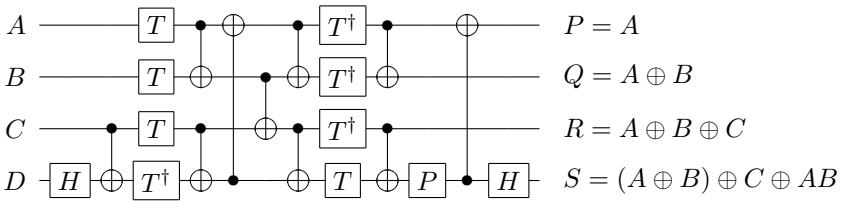
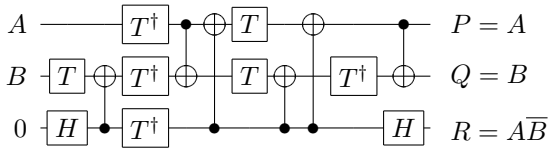


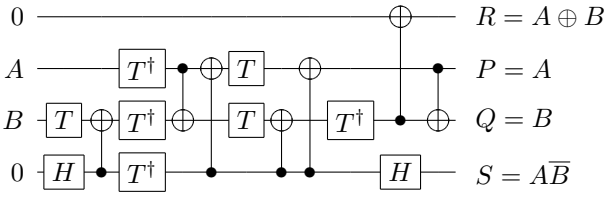
Fig. 19: Circuit implementing a reversible 1-bit Reversible Full Adder.

6.3. Reversible Half Subtractor and Reversible Full Subtractor

We use the fault-tolerant design of Controlled-V and Controlled-$V^\dagger$ gates to incorporate RFS and RHS gates. After eliminating the gates which cancel each other when they are applied to the same qubit, we have the fault-tolerant designs of RFS and RHS gates which are shown in Fig. 20 and Fig. 21.



(a) RHS1 gate corresponds to Fig. 6 with T-depth 4



(b) RHS2 gate corresponds to Fig. 7 with T-depth 4

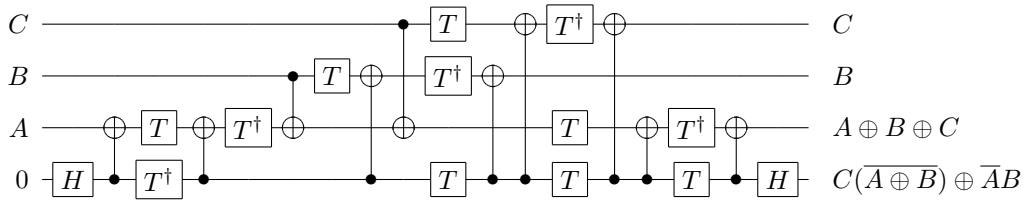
Fig. 20: Our proposed designs for Fault-Tolerant Reversible Half Subtractor.

6.4. Metrics for Fault-Tolerant Quantum Circuit

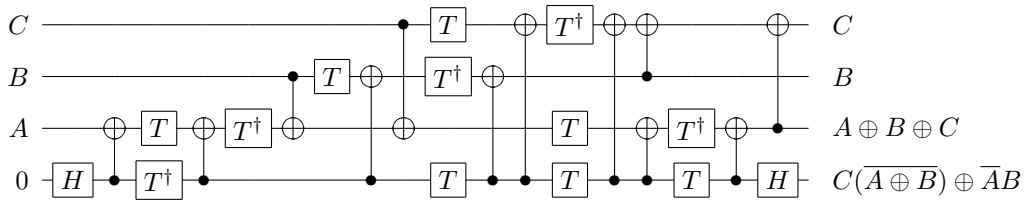
In the previous section we evaluated our proposed design in term of quantum cost, garbage output and constant cost in order to directly compare it with prior work. However, in quantum computing we often use KQ [Steane 2003] as the cost metric, which helps to calculate the demands on quantum error correction. KQ is calculated by multiplying the number of qubits used and the circuit depth, or number of time steps. Here we use the fault-tolerant design, thus we use T-depth as the circuit's depth.

Table IV shows the T-depth of each stage in the reversible floating point adder. Therefore, we can evaluation of our proposed design in term of KQ. Note that this total depth is calculated after making some parts run in parallel.

The total KQ for the whole architecture is 723,301. This compares to a KQ for a 32-bit CDKM ripple-carry adder [Cuccaro et al. 2004] of 12,474. A floating-point addition is thus nearly sixty times as expensive as fixed-point.



(a) RFS1 gate corresponds to Fig. 8 with T-depth 6



(b) RFS2 gate corresponds to Fig. 9 with T-depth 6

Fig. 21: Our proposed designs for Fault-Tolerant Reversible Full Subtractor.

Stage	T**-depth**
Swap	174
Alignment	194
Addition	57
Conversion	212
Normalization	244
Rounding	0
Total	881

Table IV: T-depth of each step in the whole architecture.

7. DISCUSSION AND CONCLUSION

With improvements in the two most hardware-intensive parts, this proposed design has reduced the Quantum Cost by 68%, the Garbage Output by 72% and the Constant Input by 71.5%. We also give a fault-tolerant version of the whole architecture.

At this stage of the execution, the system state corresponds to $\langle A', B', f(A, B), G\rangle$ where Table III has included A', B' and G under "garbage output". To complete the reversibility of the circuit, we must bring in an additional 32-bit register, execute transverse CNOTs from the output value, then run our complete circuit in reverse to clean up all of the garbage as shown in Eq. (1). Thus, the complete circuit uses 821 qubits: 64 variable input qubits and 757 input ancillae. On output, as noted, ancillae are returned to their pristine state, but 32 have been drafted into permanent use. We conclude that floating point addition is not a "green" operation, unsustainable with repeated use.

In future work, we plan to investigate restricting the ranges of input values to determine if the reversibility can be improved.

REFERENCES

M. Amy, D. Maslov, M Mosca, and M. Roetteler. 2013. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *arXiv:quant-ph/1206.0758v3* (January 2013).

A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J.A. Smolin, and H. Weinfurter. 1995. Elementary gates for quantum computation. *Physical Review A* 52, 5 (1995), 3457.

David Beckman, Amalavoyal N. Chari, Srikrishna Devabhaktuni, and John Preskill. 1996. Efficient networks for quantum factoring. *Phys. Rev. A* 54 (1996), 1034–1063. http://arXiv.org/quant-ph/9602016.

Katherine L. Brown, William J. Munro, and Vivien M. Kendon. 2010. Using Quantum Computers for Quantum Simulation. *Entropy* 12, 11 (2010), 2268–2307. DOI: http://dx.doi.org/10.3390/e12112268

Byung-Soo Choi and Rodney Van Meter. 2012. A $\Theta(\sqrt{n})$-depth Quantum Adder on a 2D NTC Quantum Computer Architecture. 8, 3, Article 24 (Aug. 2012), 22 pages. DOI: http://dx.doi.org/10.1145/2287696.2287707

BD Clader, BC Jacobs, and CR Sprouse. 2013. Quantum algorithm to calculate electromagnetic scattering cross sections. *arXiv preprint arXiv:1301.2340* (2013).

Steve A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. 2004. A new quantum ripple-carry addition circuit. http://arxiv.org/abs/quant-ph/0410184. (Oct. 2004).

Simon J Devitt, William J Munro, and Kae Nemoto. 2013. Quantum error correction for beginners. *Reports on Progress in Physics* 76, 7 (2013), 076001. http://stacks.iop.org/0034-4885/76/i=7/a=076001

Thomas G. Draper, Samuel A. Kutin, Eric M. Rains, and Krysta M. Svore. 2006. A Logarithmic-Depth Quantum Carry-Lookahead Adder. 6, 4&5 (July 2006), 351–369.

E. Fredkin and T. Toffoli. 1982. Conservative logic. *International Journal of Theoretical Physics* 21 (1982), 219253.

Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. 2008a. Architectures for a quantum random access memory. *Phys. Rev. A* 78 (Nov 2008), 052310. Issue 5. DOI: http://dx.doi.org/10.1103/PhysRevA.78.052310

Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. 2008b. Quantum Random Access Memory. *Phys. Rev. Lett.* 100 (Apr 2008), 160501. Issue 16. DOI: http://dx.doi.org/10.1103/PhysRevLett.100.160501

S. Gorgin and A. Kaivani. 2007. Reversible Barrel Shifters. In *IEEE/ACS International Conference on Computer Systems and Applications AICCSA '07*. 479–483.

M. Haghparast, S. Jassbi, K. Navi, and O. Hashemipour. 2008. Design of a novel reversible multiplier circuit using HNG gate in nanotechnology. *World Applied Sciences Journal* 3, 6 (2008), 974978.

Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. 2009. Quantum Algorithm for Linear Systems of Equations. *Phys. Rev. Lett.* 103, 15 (Oct 2009), 150502. DOI: http://dx.doi.org/10.1103/PhysRevLett.103.150502

I. Hashmi and H.M.H. Babu. 2010. An Efficient Design of a Reversible Barrel Shifter. In *23rd International Conference on VLSI Design*. 93–98. VLSID '10.

Stephen P. Jordan, Keith S. M. Lee, and John Preskill. 2012. Quantum algorithms for quantum field theories. *Science* 336 (June 2012), 1130.

Richard Jozsa. 2003. Notes on Hallgren's efficient quantum algorithm for solving Pell's equation. *arXiv preprint quant-ph/0302134* (2003).

Ivan Kassal, James D. Whitfield, Alejandro Perdomo-Ortiz, Man-Hong Yung, and Alań Aspuru-Guzik. 2011. Simulating Chemistry Using Quantum Computers. *Annual Review of Physical Chemistry* 62, 1 (2011), 185–207.

R. Landauer. 1961. Irreversibility and heat generation in the computational process. *IBM Journal of Research and Development* 5 (August 1961), 183–191.

Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. 2013. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411* (2013).

K. Matsumoto and K. Amano. 2008. Representation of Quantum Circuits with Clifford and $\frac{\pi}{8}$ Gates. *arXiv:quant-ph/0806.3834* (Jun 2008).

M. Nachtigal, H. Thapliyal, and N. Ranganathan. 2011. Design of a Reversible Floating-Point Adder Architecture. In *11th IEEE International Conference on Nanotechnology*. Portland Marriott, Portland, Oregon, USA.

A. Peres. 1985. Reversible logic and quantum computers. *Phys. Rev. A, Gen. Phys* 32, 6 (Dec 1985), 32663276.

John Preskill. 1997. Fault-tolerant quantum computation. *arXiv:quant-ph/9712048v1* (19 Dec 1997).

R. Raussendorf. 2012. Key ideas in quantum error correction. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 370, 1975 (2012), 4541–4565.

Andrew M. Steane. 2003. Overhead and noise threshold of fault-tolerant quantum error correction. *Phys. Rev. A* 68, 4 (2003), 042322–042322.

Y. Takahashi. 2009. Quantum arithmetic circuits: A survey. *IEICE Trans. Fund. Electron. Comm. Comput. Sci. E92. A* 5 (2009), 1276–1283.

Y. Takahashi and N. Kunihiro. 2005. A linear-size quantum circuit for addition with no ancillary qubits. 5, 6 (2005), 440–448.

B. M. Terhal. 2013. Quantum error correction for quantum memories. *arXiv:1302.3428 [quant-ph]* (2013).

Himanshu Thapliyal and Nagarajan Ranganathan. 2009. Design of Efficient Reversible Binary Subtractors Based on A New Reversible Gate. In *IEEE Computer Society Annual Symposium on VLSI*.

Himanshu Thapliyal and Nagarajan Ranganathan. 2011. A New Design of The Reversible Subtractor Circuit. In *11th IEEE International Conference on Nanotechnology*. Portland Marriott, Portland, Oregon, USA.

Rodney Van Meter and Kohei M. Itoh. 2005. Fast Quantum Modular Exponentiation. 71, 5 (May 2005), 052320.

Vlatko Vedral, Adriano Barenco, and Artur Ekert. 1996. Quantum networks for elementary arithmetic operations. *Phys. Rev. A* 54 (1996), 147–153. http://arXiv.org/quant-ph/9511018.

Wikipedia. Single-precision floating-point format. http://en.wikipedia.org/wiki/Single-precision_floating-point_format. (????).

Received February 2007; revised March 2009; accepted June 2009