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

Trung Duc Nguyen and Rodney Van Meter
Keio University, Faculty of Environment and Information Studies
5322 Endo, Fujisawa, Kanagawa, Japan

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. [13]. 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 designs for the circuits. 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.

I. 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 [11]. This means reversible logic has applications in low power computing. Additionally, quantum computing inherently uses reversible computing because most operations in quantum computing are unitary and work as reversible functions. For these reasons, reversible computing is receiving a lot of attention from quantum researchers.

Existing quantum algorithms generally operate on fixed-point numbers, but floating point arithmetic would likely benefit some algorithms [3, 8, 10]. While many fixed point adder designs have been introduced, 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 [13]. 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's 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.

II. BACKGROUND

A. Metrics for Evaluating Quantum Circuits

There are various methods to evaluate a circuit but in this paper we adapt 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 noted 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 [17] which we use rather than the Peres gate [14] that Nachtigal favors, while Fig. 1(b) shows the quantum Barenco decomposition [2]. 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:

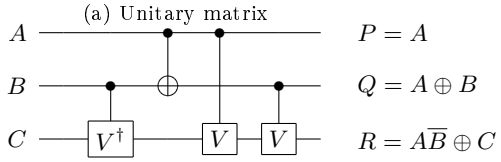
$$VV = V^\dagger V^\dagger = X,$$
$$VV^\dagger = V^\dagger V = I,$$

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

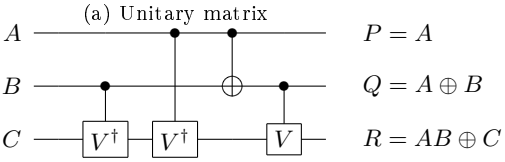


(b) Barenco decomposition

FIG. 1: TR gate

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

(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 [15], we often use another set of gates to build up circuits. The most commonly used set of gates sufficient for universal fault-tolerant quantum computation is the Clifford+T set [12]. 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. The T gate is actually a phase shift gate which 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 [1]. In this circuit, the T-depth is 4.

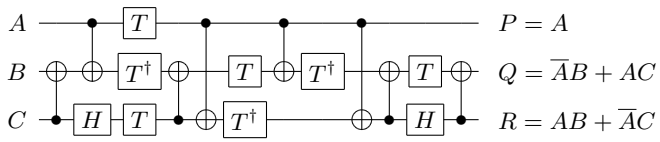


FIG. 3: Decomposition of Fault-Tolerant Fredkin gate.

B. IEEE-754 Floating-Point

The IEEE-754 single-precision floating-point format called binary32 [19] 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 stored with 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.

III. 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 numbers 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.

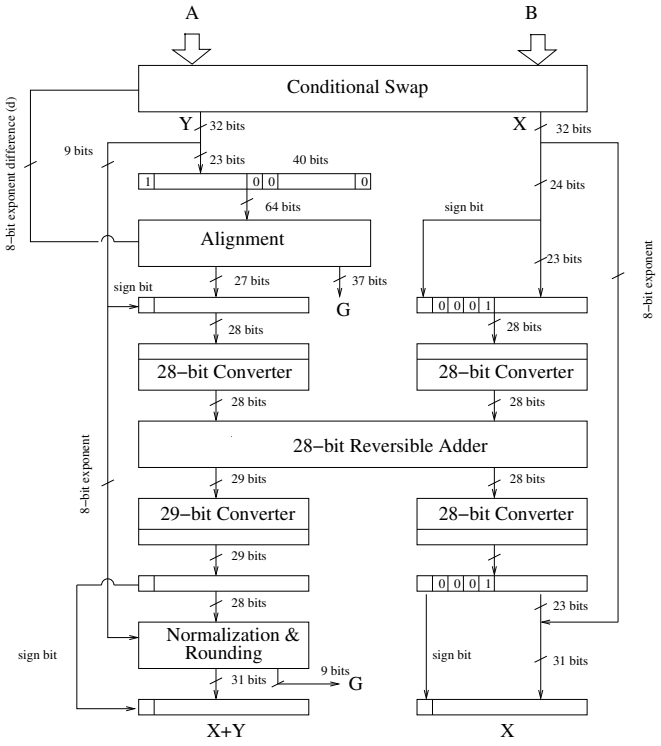


FIG. 4: Overview of the algorithm for a Floating Point Adder.

Reversible Conditional Swap. A reversible conditional swap is necessary because we need to figure out which number has the smaller exponent and then input it to the reversible alignment step. If expA < expB (expA stands for the exponent of A) then swap the two numbers, otherwise 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 [6] 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. The proposed converter will be described later.

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.

IV. 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 quantum cost respectively. Our proposed design focuses mainly on these parts, and improves some other parts in smaller ways.

A. 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 [18] 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 note 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.

B. 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 [9].

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 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.

C. 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 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.

In the conversion step, we use a total four converters: two *28-bit sign-magnitude to two's complement* converters for pre-addtion, 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.

D. 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 by 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 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 [6].

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 $O_2O_1O_0$.

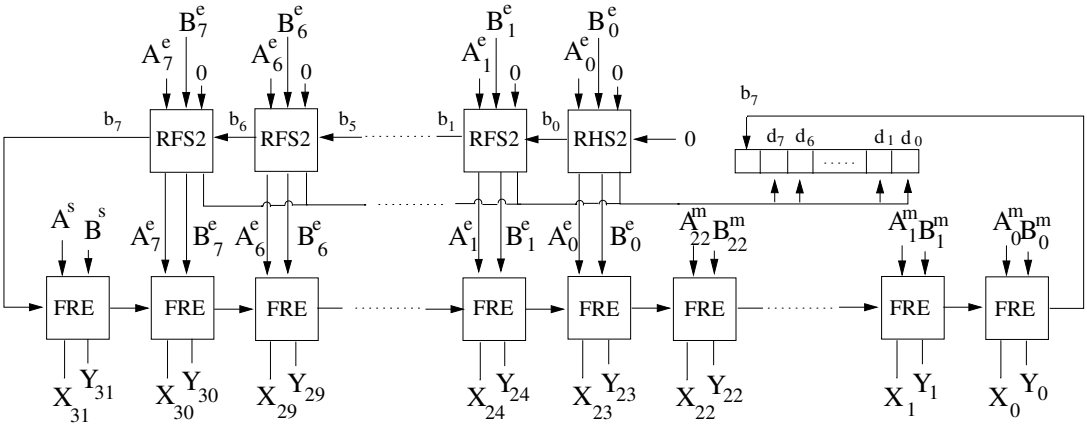


FIG. 5: Proposed Conditional Swap. The output register d is the difference of the exponents, which later will be fed into the shift limiter.

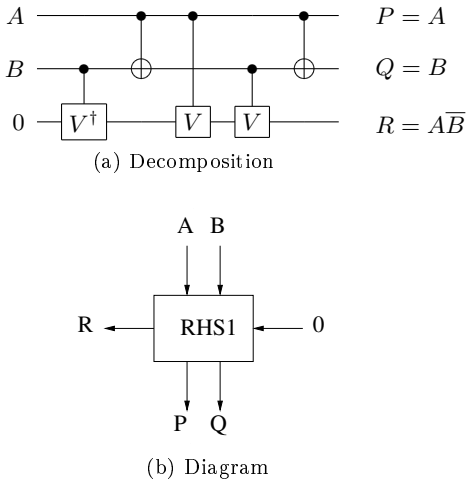


FIG. 6: Diagram and decomposition of the existing Reversible Half Subtractor (RHS1).

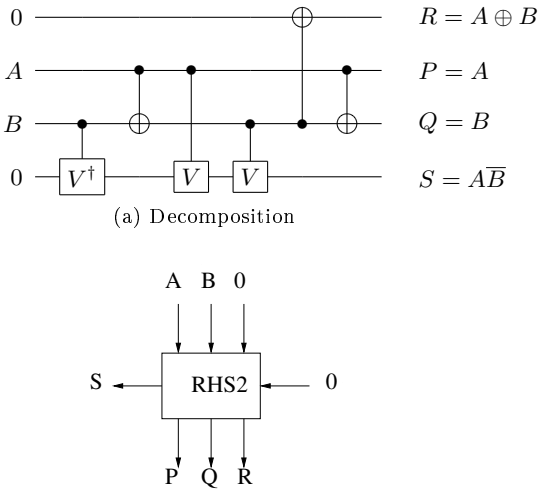


FIG. 7: Diagram and decomposition of our proposed Reversible Half Subtractor (RHS2).

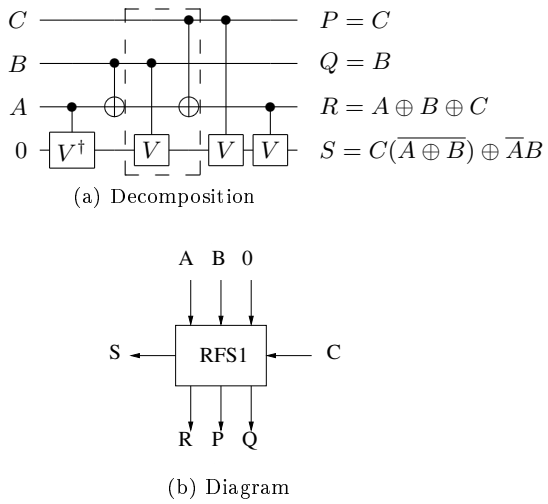


FIG. 8: Diagram and decomposition of the existing Reversible Full Subtractor (RFS1).

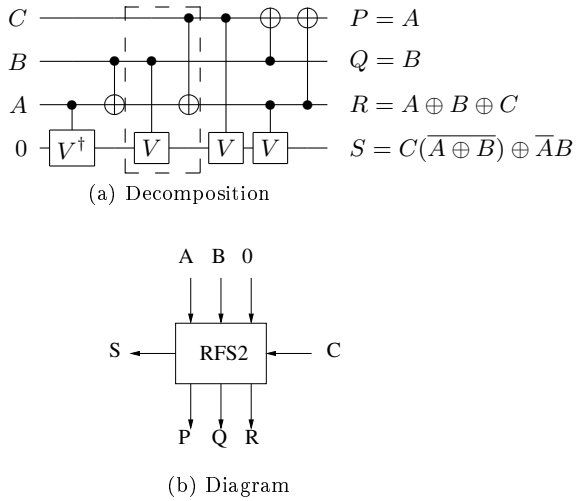


FIG. 9: Diagram and decomposition of our proposed Reversible Full Subtractor (RFS2).

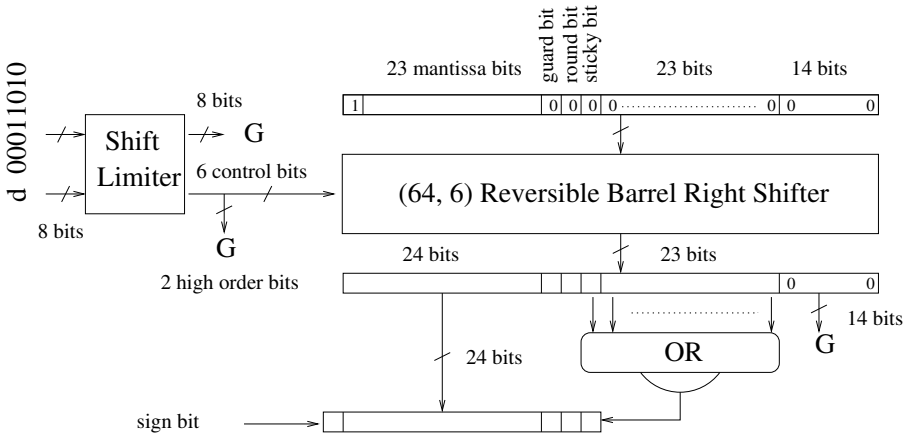


FIG. 10: Proposed design for Reversible Alignment Unit. The constant value in the top left input is the number 26, the upper limit for our shift distance. d is the exponent's difference.

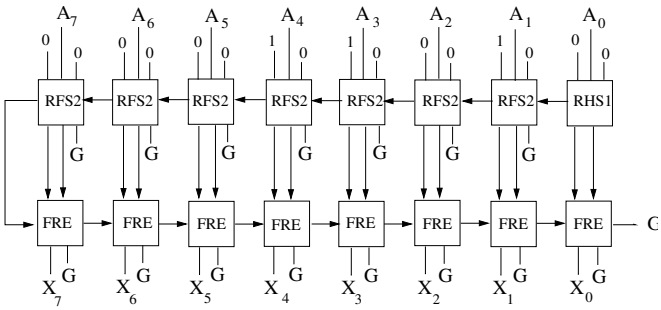
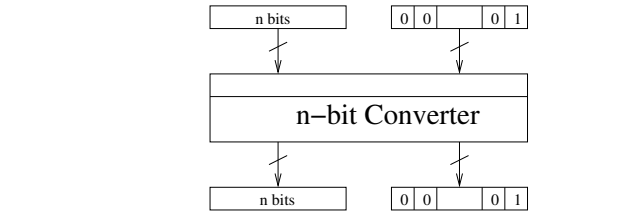
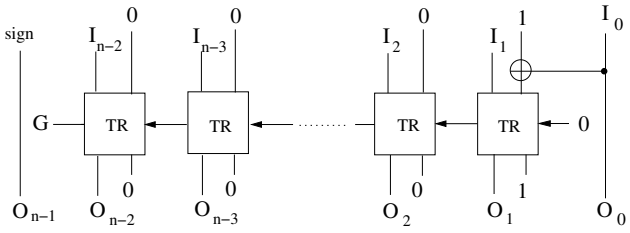


FIG. 11: The shift limiter works as a reversible conditional swap.

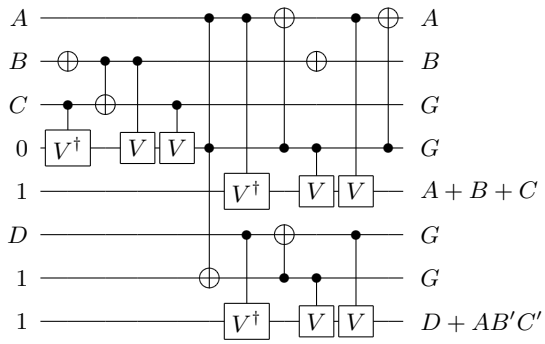


(a) Diagram

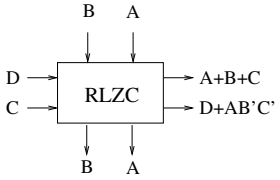


(b) Proposed Converter in details

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



(a) Decomposition



(b) Diagram

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

V. 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.

A. Reversible Conditional Swap

In the NTR design, the authors proposed using 9 HNG gates [7] as a reversible subtractor. The actual swap is done by a bank of 32 Fredkin gates [5] 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

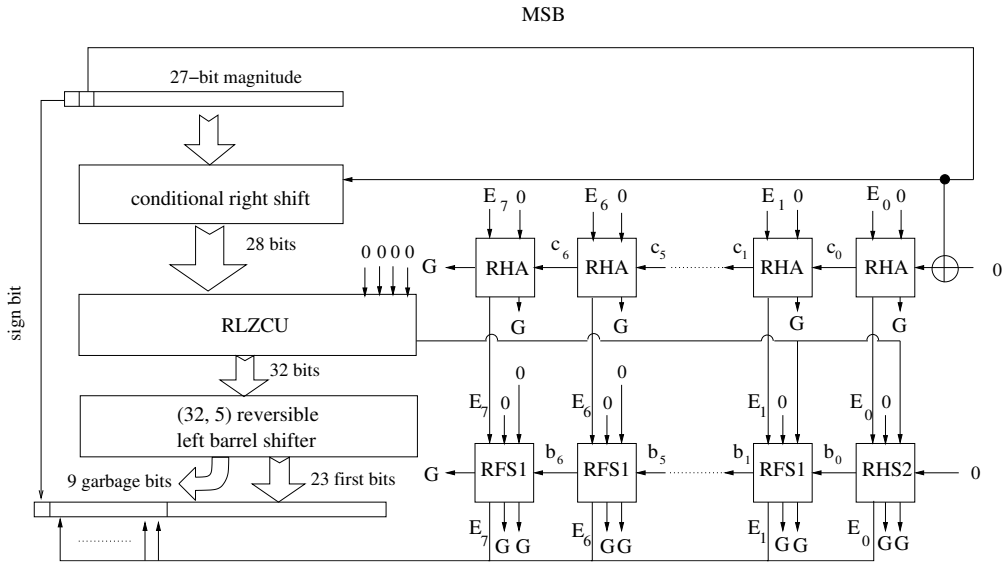


FIG. 14: Reversible Normalization.

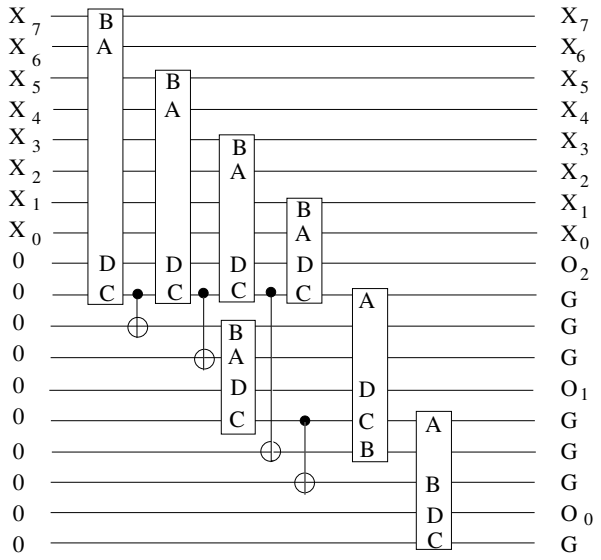


FIG. 15: An example of using RLZCU for 8 input bits.

subtractor. The evaluation and comparison is shown in Table II.

B. Reversible Alignment

A (256, 8) reversible barrel shifter (256 input bits, 8 control lines) [9] is used in the NTR design. This is one of the key reasons for the large quantum cost, constant input and garbage output 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 [9] the number of Fredkin gates and Feynman 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.

C. Reversible Converter

The NTR design used $n-1$ Peres gates [14] 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 conversions.

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

	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.

	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.

that problem.

D. 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.

E. 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.

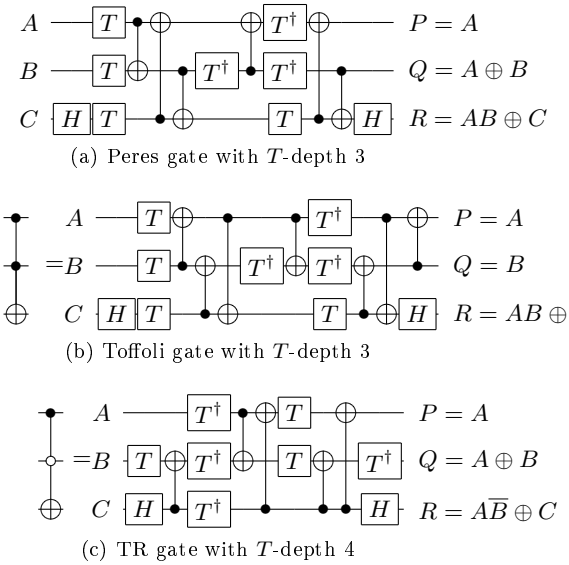
VI. FAULT-TOLERANT DESIGN

In this paper, we have seen 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. [1] 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.

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.



(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

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.

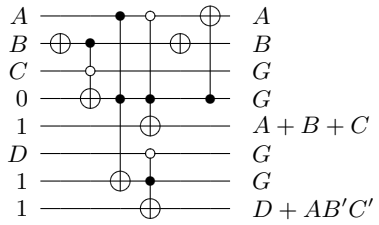


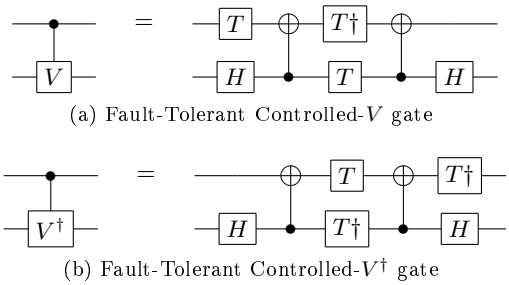
FIG. 17: Fault-Tolerant RLZC circuit

A. 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 [1] 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.

B. Reversible Full Adder

For our 28-bit adder, we use 27 RFA gates (Reversible Full Adder) and a one RHA gate, which is simply imple-



(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

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.

mented by a single Peres gate. Fig. 19 shows the decomposition of RFA gate described in [1]. This RFA gate has T-depth of 2.

C. 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.

D. 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 [16] 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 [4] of 12,474. A floating-point addition is thus nearly sixty times as expensive as fixed-point.

VII. 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.

Acknowledgments.

This work was supported by the Japan Society for the Promotion of Science (JSPS) through its "Funding Program for World-Leading Innovative R&D on Science and Technology (FIRST Program)".

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

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

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

[4] Steve A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit. http://arxiv.org/abs/quant-

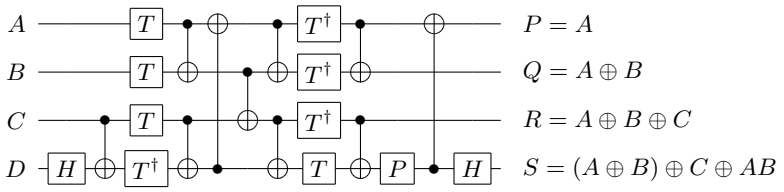
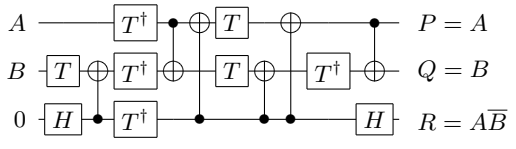
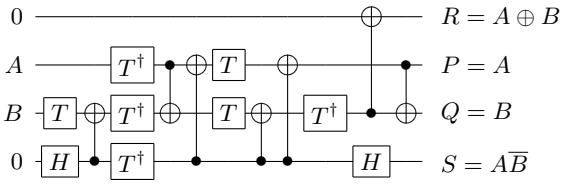


FIG. 19: Circuit implementing a reversible 1-bit Reversible Full Adder.



(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.

ph/0410184, October 2004.

[5] E. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21:219–253, 1982.

[6] S. Gorgin and A. Kaivani. Reversible barrel shifters. In *IEEE/ACS International Conference on Computer Systems and Applications AICCSA '07*, pages 479–483, May 2007.

[7] M. Haghparast, S. Jassbi, K. Navi, and O. Hashemipour. Design of a novel reversible multiplier circuit using hng gate in nanotechnology. *World Applied Sciences Journal*, 3(6):974–978, 2008.

[8] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103(15):150502, Oct 2009.

[9] I. Hashmi and H.M.H. Babu. An efficient design of a reversible barrel shifter. In *23rd International Conference on VLSI Design*, pages 93–98, 3-7 Jan 2010. VLSID '10.

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

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

[12] K. Matsumoto and K. Amano. Representation of quantum circuits with clifford and $\frac{\pi}{8}$ gates. *arXiv:quant-ph/0806.3834*, Jun 2008.

[13] M. Nachtigal, H. Thapliyal, and N. Ranganathan. Design of a reversible floating-point adder architecture. In *11th IEEE International Conference on Nanotechnology*, Portland Marriott, Portland, Oregon, USA, August 2011.

[14] A. Peres. Reversible logic and quantum computers. *Phys. Rev. A, Gen. Phys*, 32(6):3266–3276, Dec 1985.

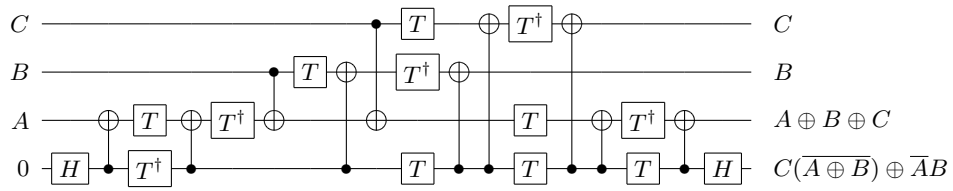
[15] John Preskill. Fault-tolerant quantum computation. *arXiv:quant-ph/9712048v1*, Dec 1997.

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

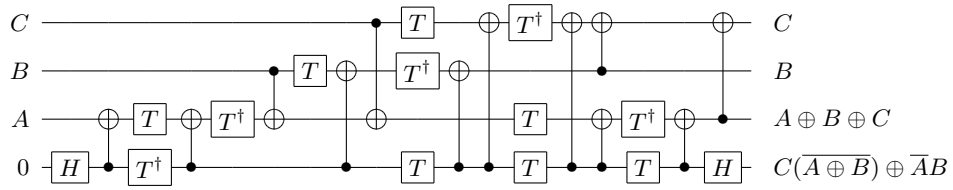
[17] Himanshu Thapliyal and Nagarajan Ranganathan. Design of efficient reversible binary subtractors based on a new reversible gate. In *IEEE Computer Society Annual Symposium on VLSI*, 2009.

[18] Himanshu Thapliyal and Nagarajan Ranganathan. A new design of the reversible subtractor circuit. In *11th IEEE International Conference on Nanotechnology*, Portland Marriott, Portland, Oregon, USA, 15-18 August 2011.

[19] Wikipedia. Single-precision floating-point format. `http://en.wikipedia.org/wiki/Single-precision_floating-point_format`.



(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.