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

Trung Duc Nguyen and Rodney Van Meter
Faculty of Environment and Information Studies
Keio University, SFC

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. [11]. 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% compared to the existing design. We also propose fault-tolerant designs for the architectures. 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

A. Existing Reversible Floating-Point Adder and Our Proposed Design

Computer arithmetic is generally carried out as either integer arithmetic, more correctly called fixed-point arithmetic in most contexts, or floating point arithmetic. Floating point numbers, as the name implies, allow the decimal (binary) point to be repositioned according to the value of an exponent. Fixed point numbers are limited in general to the range $-2^n..2^n-1$ for an n-bit number, but a floating point can cover many orders of magnitude larger or smaller values by adjusting the exponent. The price for this flexibility is reduced precision within the same storage space, as some bits are dedicated to storing the exponent, and substantially higher execution costs. However, floating point is the standard representation for almost all scientific data, as data points often span a broad dynamic range or a range that is difficult to determine *a priori*.

Some quantum computing algorithms would benefit from the availability of a library of floating point operations, if one were available. Algorithms that focus on physical phenomena, such as quantum chemistry [1, 9] and quantum field theory calculations [7], seem especially likely to be able to take advantage of this capability. Implementations of Harrow, Hassidim and Lloyd's

algorithm for linear systems could substantially broaden their applicability [2, 5]. Jozsa's variant of Hallgren's algorithm that uses real numbers also seems to be a likely candidate, with further adaptation [4, 8].

While many fixed point adder designs have been introduced, we are aware of only one design for a floating-point quantum 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 [11].

There are various methods to evaluate a circuit but we will first 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.

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.

Table I shows the total comparison between our proposed design and NTR design.

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.

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 I: NTR Design vs Proposed Design.

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 $f(A, B)$. Nachtigal'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,

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

to complete the reversibility and make the circuit suitable for quantum computation.

B. Fault-Tolerant Architecture

The NTR design is designed ambigously and constructed from the gates which make the circuit hard to implement in quantum computing. Thus, we propose a fault-tolerant design for the whole architecture.

In fault-tolerant quantum computation [12], 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 [10]. Because T or $T^\dagger$ gate are the most expensive elements in Clifford+T set. 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, to calculate the depth of the circuit.

In the previous subsection we evaluated our proposed design in term of quantum cost, garbage output and constant cost as above in order to directly compare it with prior work. However, in quantum computing we often use KQ [13] 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 fault-tolerant design, thus we use T-depth as the circuit's depth.

Table II shows the T-depth of every 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.

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

TABLE II: T-depth of each step in the whole architecture.

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

II. FLOATING-POINT ADDITION ALGORITHM

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. 1 shows the gen-

eral algorithm adapted to show constant inputs and garbage outputs. The garbage outputs are eventually cleaned by reversing this circuit using Bennett's method.

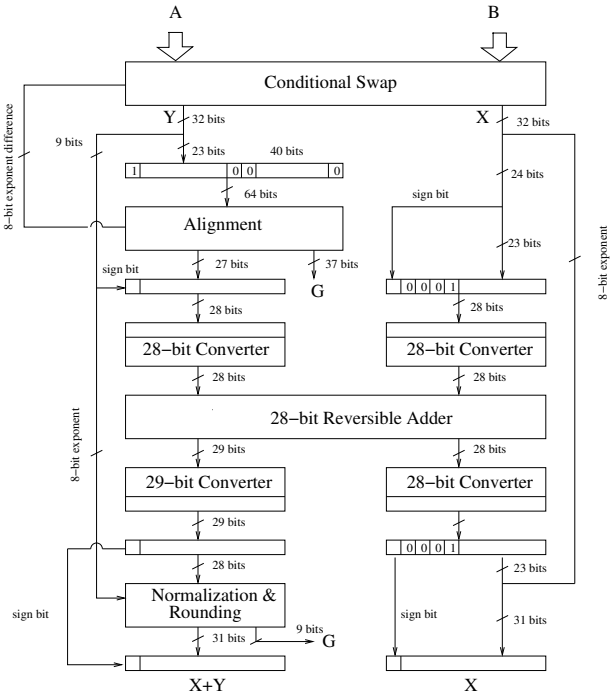

FIG. 1: Overview of the algorithm for a Floating Point Adder.

A. 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 [3] in the next step.

B. Reversible Alignment

We need reversible alignment because we can only add two mantissas when the two exponents are equal, so we need a reversible shifter. 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. Thus, a (256, 8) reversible barrel shifter (256 input bits, 8 control lines) [6] 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. Our design significantly reduces this cost.

The IEEE-754 specification also requires three 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.

C. Two's complement Conversion and Reversible Addition

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. 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.

D. 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 need 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.

III. DISCUSSION AND CONCLUSION

With improvements in the two most hardware-intensive parts, this proposed design has reduced the quantum cost by 68%, the number of garbage outputs by 72% and the number of constant inputs 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 I 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.

[1] Katherine L. Brown, William J. Munro, and Vivien M. Kendon. Using quantum computers for quantum simulation. *Entropy*, 12(11):2268–2307, 2010.

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

[3] 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.

[4] S. Hallgren. Polynomial-time quantum algorithms for Pell's equation and the principal ideal

problem. *Journal of the ACM (JACM)*, 54(1), 2007.

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

[6] 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.

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

[8] Richard Jozsa. Notes on hallgren's efficient quantum algorithm for solving pell's equation. *arXiv preprint quant-ph/0302134*, 2003.

[9] Ivan Kassal, James D. Whitfield, Alejandro Perdomo-Ortiz, Man-Hong Yung, and Alań Aspuru-Guzik. Simulating chemistry using quantum computers. *Annual Review of Physical Chemistry*, 62(1):185–207, 2011.

[10] K. Matsumoto and K. Amano. Representation of quantum circuits with clifford and $\frac{\pi}{8}$ gates.

[11] 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.

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

[13] Andrew M. Steane. Overhead and noise threshold of fault-tolerant quantum error correction. *Phys. Rev. A*, 2003.