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

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. [?]. 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, we have reduced the cost about 68% compared to the existing design. We also propose fault-tolerant 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.

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

TABLE I: T-depth of each stage in our fault-tolerant adder.

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 to the range $-2^n..2^n-1$ for an n-bit number, but a floating point can cover many more orders of magnitude. 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 scientific data, as data points often span a broad dynamic range or a range that is difficult to determine *a priori*.

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 [? ?] and quantum field theory calculations [?], seem especially likely to be able to take advantage of this capability. Implementations of Harrow, Hassidim and Lloyd's algorithm for linear systems [? ?] and Jozsa's variant of Hallgren's algorithm that uses real numbers may also benefit [?].

While many fixed point adder designs have been introduced, we are aware of only one design for a reversible floating-point quantum adder, by Nachtigal, Thapliyal and Ranganathan (NTR), and this design is expensive [?]. Our proposed design eliminates about 68% of the cost. 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 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.

This 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.

In quantum computing we often use KQ [?] 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. Table ?? shows the T-depth of each stage. 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.

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. 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. ?? 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.

At this stage of the execution, the system state corresponds to $\langle A', B', f(A,B), G\rangle$. 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, most ancillae are returned to their pristine state,

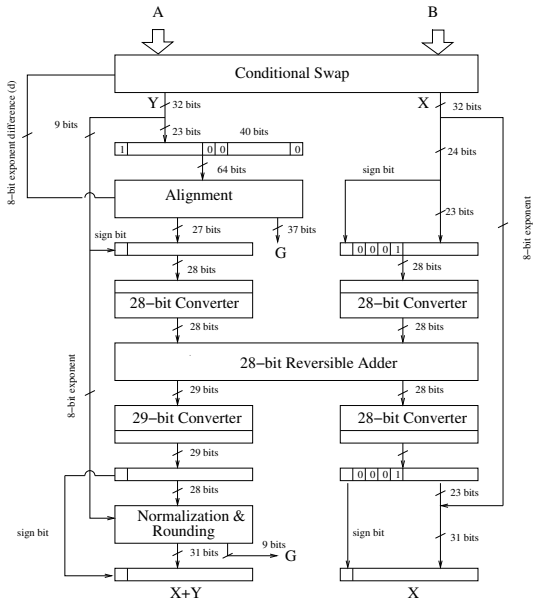

FIG. 1: Overview of our floating point adder.

but 32 have been drafted into permanent use. We conclude that floating point addition is not a “green" operation, unsustainable with repeated use.