Floating Point Representations in Quantum Circuit Synthesis

Nathan Wiebe[1,2], Vadym Kliuchnikov[1,3]
[1] *Institute for Quantum Computing, 200 University Ave. West, Waterloo, ON, Canada*
[2] *Department of Combinatics & Opt., University of Waterloo, Waterloo, ON, Canada and*
[3] *Department of Computer Science, University of Waterloo, Waterloo, ON, Canada*

We provide a non–deterministic quantum protocol that approximates $R_x(\phi_1^2\phi_2^2)$ using $R_x(\phi_1)$ and $R_x(\phi_2)$ and a constant number of Clifford and T operations. We then use this method to construct a "floating point" implementation of a small rotation wherein we use the aforementioned method to construct the exponent part of the rotation and also to combine it with a mantissa. This causes the cost of the synthesis to depend more strongly on the relative (rather than absolute) precision required. We analyze the mean and variance of the T–count required to use our techniques and show that, with high probability, the required T–count will be lower than lower bounds for the T–count required to do ancilla–free circuit synthesis. We also discuss the T–depth of our method and show that the vast majority of the cost of the resultant circuits can be shifted offline.

The ability to inexpensively perform single–qubit rotations is vital for quantum computing. These rotations form the core of several quantum algorithms including the quantum Fourier transform, quantum simulation and the synthesis of multi–qubit unitaries, among many others. Optimizing the cost of synthesizing single–qubit rotations is vitally important for designing practical algorithms on the first generation of fault–tolerant quantum computers. Until very recently, the Solovay–Kitaev theorem [1] was the best known technique for synthesizing such rotations. A revolution has occurred in the last several months in the field of circuit synthesis, providing a polynomial improvement over the Solovay–Kitaev theorem, causing the cost of circuit synthesis to approach information theoretic bounds for the optimal scaling [2–4]. The question that remains is: "how efficient can we make the task of synthesizing single–qubit rotations?"

We address this question by providing a new paradigm for synthesizing small single–qubit rotations that is not only more efficient than existing methods, but is also more efficient than *any method* that does not use ancillas to assist the synthesis. The key insight behind this method is that a floating point representation of the rotation angle can be used to simplify the synthesis of small rotations. Traditional circuit synthesis methods, in effect, treat every leading zero in a decimal representation of a small rotation as a significant digit. This makes synthesizing these rotations costly in cases where a small rotation angle is needed, but the number of digits of precision that are required of the rotation is small. Our floating point representation solves this problem by providing a way to multiply small rotation angles, which addresses the issue of extraneous digits of precision for circuit synthesis in exactly the same way that floating point representations of decimal numbers remove the need to keep track of irrelevant digits of precision in arithmetic problems.

There are two central components to our method. The first component is a non–deterministic circuit that can combine two rotations $R_x(\alpha)$ and $R_x(\beta)$ to approximate $R_x(\alpha^2\beta^2)$. The second component is a non–deterministic circuit that can inexpensively generate small X–rotations. The intuition behind how these two components combine to form our floating point representation is demonstrated in the following example. Imagine that we want to implement the rotation $R_x(a \times b^{-\gamma})$, where $b \approx 0.029$, γ is an integer and $1 > a > b$. This rotation can be approximated using the first component of our method to combine the rotations $R_x(\sqrt{a})$ and $R_x(b^{-\gamma/2})$, which we call the mantissa and exponent unitaries respectively. The mantissa unitary can be synthesized inexpensively using traditional circuit synthesis methods because it requires relatively few digits of precision; whereas the exponent unitary is implemented using the non–deterministic circuits that are the second component of our method.

It should be noted that although our circuits are non–deterministic, any failures that occur can be corrected using Clifford operations. This means that our circuits will always succeed, although the total cost of implementing our circuits will vary. We show that, despite the uncertainty in the cost of synthesizing the rotation, our floating point method will be less expensive than the best known techniques with high probability.

We consider three different scenarios in which to assess the cost of our algorithm. In all three scenarios, we assume that the most expensive gate in the $\{\text{Clifford}, T\}$ gate library is the T gate and assume for simplicity that Clifford operations are free. In the first scenario, we measure the cost by counting the total number of T gates required to synthesize the rotation. This cost analysis is appropriate in situations where a serial quantum computer is used to execute the floating point circuits. The second scenario assumes that the quantum computer is massively parallel and hence the T–depth of the circuit best represents the time required to execute the circuit. Finally, we consider the online cost of our algorithm, which is an appropriate measure in cases where "factories" can be employed that constantly produce resource states that can be consumed cheaply throughout the protocol. All three scenarios show that our approach has substantial advantages over the best known synthesis methods.

The reduction in the T–count is perhaps the strongest result that arises from our method. We show that, on

average, floating point synthesis of a small rotation θ using a fixed number of digits of precision requires a number of T gates that approximately scales as $1.14\log_2(1/\theta)$. This is significant because an information theoretic lower bound gives that $O(\log_2(1/\theta))$ scaling is optimal for the T–count required to synthesize a small rotation [5]. Apart from achieving optimal scaling, we actually can show that floating point synthesis is superior to the best possible circuit synthesis method that only uses single–qubit Clifford at T gates, which we show requires a number of T gates that scales approximately as $2.98\log_2(1/\theta)$. To the best of our knowledge, this is the first conclusive demonstration that synthesis methods that use ancillas and classical feedback are more powerful those that do not.

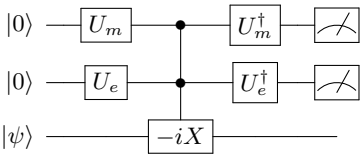
As a particular example of the performance of our method, we compare the cost of synthesizing the rotation $e^{-iZ\pi/2^{16}}$ (which is used in the quantum Fourier transform) using the $\{\text{Clifford}, T\}$ gate library with our floating point synthesis method to the best possible method for synthesizing the rotation using only single–qubit Clifford and T gates.

Mantissa Unitary (Floating Point Synthesis)	Mean T–count	Variance	95% Confidence Interval	Relative Error
$HZTHZTHZTH$	24.2	11.8	[21,33]	0.35
$HTHTHTHTHTHTH$	30.3	14.0	[27,39]	0.13
Circuit (Optimal single–qubit Synthesis)	**T–count**	**–**	**–**	**Relative Error**
Non–Trivial Approximation With Smallest T–Count	57			0.17
Non–Trivial Approximation With 2^{nd} Smallest T–Count	60			0.058

This shows that floating point synthesis can provide advantages for synthesizing even modestly small rotations that appear in important quantum algorithms. Additionally, the classical algorithm for finding the optimal quantum circuit (drawn from the single–qubit Clifford and T library) is inefficient. This prevents the optimal synthesis method from synthesizing rotations smaller than 10^{-16} rad. Floating point synthesis suffers no such drawback.

We find that the T–depth and the online cost of floating point synthesis for $R_x(\theta)$, for a fixed number of digits of precision, scales as $\Theta(\log\log(1/\theta))$. In contrast, traditional circuit synthesis leads to T–depths and online costs that scale as $O(\text{polylog}(1/\epsilon))$, where ϵ is the absolute error tolerance and $\epsilon < \theta$. This is significant because it shows that massive parallelism can be used to exponentially speed up the implementation of these rotations, which means that performing the small rotations required in quantum simulation algorithms or Shor's algorithm on a parallel quantum computer is far cheaper than existing methods, such as the Solovay–Kitaev algorithm, would suggest.

Now that we have discussed the improvements that floating point synthesis can bring, we will now describe in greater detail how our method achieves these performance improvements. Our circuit for combining the mantissa unitary, U_m, with the exponent unitary, U_e, to form the floating point representation is surprisingly simple:



If both measurements yield zero, then the circuit will implement

$$|\psi\rangle \to e^{-i\tan^{-1}(\tan^2(\sin^{-1}(|U_{m\,1,0}||U_{e\,1,0}|)))X} \approx e^{-i|U_{m\,1,0}|^2|U_{e\,1,0}|^2X}.$$

In all other cases the circuit implements $e^{iX\pi/4}$, which can be inverted using Clifford operations. The success probability for implementing small rotations using this circuit is nearly 100%; furthermore, the cost of failure is minimal using this circuit because errors can be corrected using Clifford operations which are typically assumed to be inexpensive in circuit synthesis problems. This means that this circuit cannot possibly fail, although it may have to be applied several times before success is achieved in rare cases.

Our method for generating small rotations uses similar principles. The intuition behind the method is that it generates a small rotation by (approximately) iteratively squaring the rotation angle. To see how this works, let us assume for simplicity that a circuit is known that implements an X–rotation: $\exp(-i\theta X)$ for some value of θ. The circuit in Figure 1 (a) then, upon measurement of zero, enacts $\exp(-i\tan^{-1}(\tan^2(\theta))) \approx \exp(-i\theta^2 X)$ and failure results in the application of a Clifford operation, which can be corrected inexpensively. By using this circuit recursively d–times, it is possible to generate a rotation that is approximately $\exp(-i\theta^{2^d}X)$ when the outcome of every measurement is zero. An example of this recursively constructed circuit for $d = 3$ is given in Figure 1 (b). We take $U = HTH$, rather than choosing it to be an X–rotation. This leads to the circuit implementing, upon success,

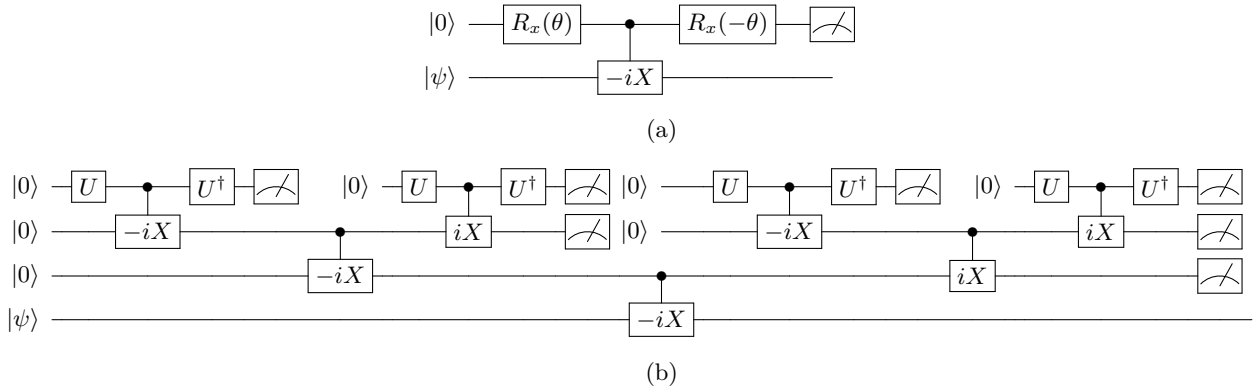


FIG. 1: Circuits for implementing small rotations by (approximate) repeated squaring of the rotation angle. (a) shows the basic circuit for recursion depth $d = 1$. (b) shows the analogous circuit for recursion depth $d = 3$, where U represents an arbitrary unitary operation (typically taken to be HTH in practice).

a rotation that is approximately $\exp(-i8.67 \times 10^{-4} X)$. Continuing the same recursive pattern to $d = 9$ results in a rotation that is smaller than 10^{-200} radians. Although the error correction process is more involved in this case, it is conceptually identical, and any fault can be corrected using Clifford operations.

The rotation angles constructed by this method with $U = HTH$ approximately scale as $\tan^{-1}(\tan^{2^d}(\pi/8))$, which means that recursing to a depth $d \in \Theta(\log\log(1/\theta))$ is necessary to achieve a rotation through an angle of size θ or smaller. This method alone is insufficient to generate the exponent unitary because it does not give precise control over the resultant rotation. We increase the precision of the exponent unitary by combining $D \in \Theta(\log\log(1/\theta))$ such rotations together to form a rotation through an angle that is approximately $\tan(\pi/8)^{4(2^{D-1})}$, for any $D \geq 1$. This allows us to construct an exponent unitary that shrinks as powers of $\tan(\pi/8)^4 \approx 0.029$, which enables the construction of a floating point rotation of the form $R_x(a \times b^{-\gamma})$, where $b \approx 0.029$ and $a \in (b, 1)$.

This work is significant because it provides a new approach to quantum circuit synthesis that is not only more efficient than existing circuit synthesis approaches for synthesizing small rotations, and because it reveals that the cost of circuit synthesis depends more strongly on the number of digits of precision required, rather than the absolute precision. Our work also shows that the use of ancillas allows rotations to be synthesized at lower cost than the best possible ancilla–free single–qubit synthesis algorithm that does and that parallelism can be exploited to exponentially reduce the time required to synthesize a given rotation. The performance improvements offered by our floating point synthesis method are especially important because they can be used to substantially reduce the costs of performing quantum simulation algorithms and Shor's algorithm fault–tolerantly, and in turn allow us to get one step closer to the dream of performing a practical quantum computation.

[1] Christopher M. Dawson and Michael A. Nielsen. The solovay-kitaev algorithm. *Quantum Info. Comput.*, 6(1):81–95, January 2006.
[2] Alex Bocharov and Krysta M. Svore. Resource-optimal single-qubit quantum circuits. *Phys. Rev. Lett.*, 109:190501, 2012.
[3] P. Selinger. Efficient Clifford+T approximation of single-qubit operators. *arXiv:1212.6253*, December 2012.
[4] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Practical approximation of single-qubit unitaries by single-qubit quantum Clifford and T circuits. *arXiv:1212.6964*, December 2012.
[5] A. W. Harrow, B. Recht, and I. L. Chuang. Efficient discrete approximations of quantum gates. *Journal of Mathematical Physics*, 43:4445–4451, September 2002.