Quantum-Kit: Simulating Shor's Factorization of 24-Bit Number on Desktop

2 authors:

Archana Tankasala
Purdue University
18 PUBLICATIONS **171** CITATIONS

Hesameddin Ilatikhameneh
Purdue University
83 PUBLICATIONS **1,703** CITATIONS

Some of the authors of this publication are also working on these related projects:

Low power electronics based on low-dimensional materials View project

Nanahub tool View project

Quantum-Kit: Simulating Shor's Factorization of 24-Bit Number on Desktop

Archana Tankasala, Hesameddin Ilatikhameneh

Abstract - **Quantum-Kit is a graphical desktop application for quantum circuit simulations. Its powerful, memory-efficient computational engine enables large-scale simulations on a desktop. The ability to design hybrid circuits, with both quantum and classical bits and controls, is employed to demonstrate Kitaev's approach to Shor's factorization algorithm. For the first time, Shor's factorization of a 24-bit integer is simulated with Quantum-Kit in a mere 26 minutes on a modest desktop with Intel Core i5 7400T, 2.4GHz and 12GB RAM, while the largest number factorized so far has been a 20-bit integer run on a supercomputer. For the conventional Shor's algorithm, Quantum-Kit is about 35 times faster than previously reported simulation times, whereas, with Kitaev's approach, Quantum-Kit is 6 orders of magnitude faster in Shor's factorization of exactly the same number.**

Quantum computing and communication are promising revolutionary technologies that exploit quantum phenomena like superposition and entanglement to achieve an exponential advantage over their classical counterparts [3-8]. Quantum computing has unlocked new applications ranging from logistics and scientific research to encryption and machine learning, currently driven dominantly by digital technologies [9-12]. It shows the potential to break secure encrypted codes and mine big-data exponentially faster than classical computers [13-14]. Several big players in the industry are already heavily invested in the race to make the world's first practical and viable quantum computer. While the quantum hardware development advances, the capability to simulate quantum algorithms is critical in understanding the performance and potential of real quantum machines when they are ready. To successfully understand and build quantum algorithms that solve highly demanding problems, it helps to run experiments on a computer, using numerical modeling and simulations, ideally emulating a real quantum machine.

Quantum-Kit (Q-Kit) is a graphical quantum circuit emulator that is useful in effortlessly constructing complex quantum circuits, analyzing them and visualizing the quantum states after every quantum operation. While there are several existing software freely available for these quantum simulations, they are either focused on learning or on large-scale simulation capabilities. The former kind are equipped with a user-friendly interface - specifically set-up for learning on fewer qubits, while the latter require a knowledge of quantum programming languages to truly exploit the potential of these software on supercomputing clusters.

Fig. 1. (a) Quantum-Kit Graphical User-Interface. Panels for (1) graphical quantum **circuit** builder, (2) quantum circuit **command** script, and (3) simulation execution **status**. (b) Visualization of **quantum state** after each quantum gate operation in the circuit. Single qubit states shown on a Bloch Sphere and multi-qubit gates on bar charts for probability distributions and amplitudes.

Q-Kit.

Q-Kit is aimed at achieving a balance of the two, a user-friendly interface with capabilities to efficiently run large simulations. Quantum circuits can be set-up in Q-Kit in two different ways. First, using graphical circuit builder, as shown in the top panel of the Q-Kit GUI in Fig. 1(a). Circuits are constructed simply by adding qubits from the menu, assigning quantum gates (single-qubit gates, entangling gates, multi-qubit gates or oracle) by clicking on the qubits, followed by (in-place or deferred) measurement of the qubits.

A second way, especially useful for large quantum circuits, is to set-up with *only* three commands to (1) add qubits, (2) perform a gate operation, and (3) measure, as shown in the bottom left editor panel in Fig. 1(a). An additional command to (4) define custom quantum gates gives users the flexibility to define a new (oracle or unitary) gate as a matrix or as a function. A quantum gate can be defined once and used across circuits. This powerful feature is extremely valuable in employing Q-Kit for research, as this eliminates any dependence on addition of necessary gates to the list of pre-defined gates in Q-Kit. Furthermore, a click-button conversion of the circuit from circuit-diagram to command-script and vice-versa, makes it easy to learn Q-Kit and to edit complex circuits quickly.

Status of the running simulations is displayed in the bottom right panel of Q-Kit, shown in Fig.1(a). A thorough logging information on the execution details, including timing analysis, is printed to this panel and saved to a file as well.

At the end of execution, not only is the final quantum state printed to this panel but the quantum state (after every gate operation in the quantum circuit) is also available in the circuit-diagram, as shown in Fig. 1(b). The quantum state can be viewed as (1) probability amplitudes with the complex coefficients, (2) normalized probability distribution, or (3) plain text for user post-processing.

In addition, Q-Kit has several features including the ability to copy data from measured qubits to classical bits and using classical controls for quantum operations. This ability to use classical

bits and controls in a quantum circuit, especially in certain algorithms, offers exponential advantages in simulations, both in terms of speed and memory requirements, as discussed in the next section. Q-Kit can be run both in performance-enhanced or memory-enhanced mode and is demonstrated to be extremely efficient although it currently runs only on a single core. A parallelized Q-Kit is expected to enable much more powerful computations, with far fewer resources.

Quantum emulators like Q-Kit are an invaluable tool in an intuitive understanding of the quantum circuits, like quantum teleportation, Shor's factorization or Grover's search algorithms. It allows experimenting different quantum algorithms by modeling qubits on a desktop or laptop computer to study the complex quantum systems on a computer instead of a laboratory. It also allows testing the quantum software on the emulator before running on quantum hardware. Q-Kit has applications not only to research, development and validation of quantum algorithms but also towards an intuitive understanding of quantum circuits, as a high-level precursor to quantum programming.

Shor's Factorization with Q-Kit.

Exponential-time factorization of large semi-prime integers is fundamental to encrypting schemes, like RSA, in securing digital information. Shor's factorization is a polynomial time algorithm that challenges this encryption and has been of great interest, especially with the recent developments in realization of quantum hardware [15-20].
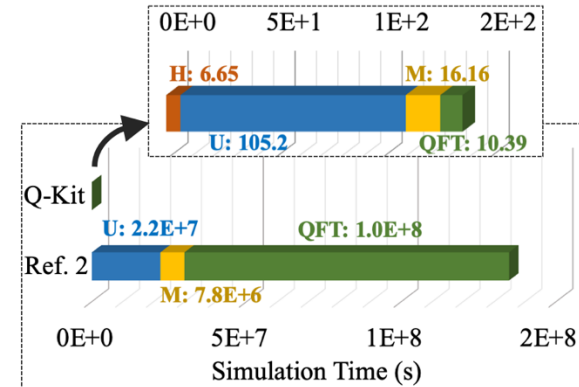

Fig. 2: Timing analysis. Q-Kit vs. Ref.[2] for Shor's factorization of 20-bit N=961307, plotted on a LOG scale. Q-Kit, with *nx2n* approach, is approximately 6 orders of magnitude faster than Ref.[2] with *3nx1* approach.

The capabilities of quantum computing simulation software are also most commonly evaluated on Shor's factorization algorithm[1-2,21-22]. While the algorithm is ingenious and offers exponential advantage to a quantum computer against its classical counterpart, classical simulation of the algorithm is extremely computationally demanding. Factorization of a 10-bit number, for example, requires 10 qubits in the work register and 20 qubits in the control register, a total of 30 qubits demanding 32MB of memory. This memory requirement scales exponentially with number of qubits, demanding as much as 32GB for a 15-bit number, clearly not feasible on conventional desktop machines.

Alternate proposals to Shor's factorization algorithm include Kitaev's version with a clever one-control-qubit trick [23-25], with 10 qubits in the work register and only a single qubit in the control register, which is "recycled" several times. This approach uses a memory of 32KB (instead of 32GB) for factorizing a 15-bit number. However, although the approach is memory efficient and requires fewer qubits, QFT is applied over several stages in the circuit (instead of just once). For the rest of this paper, the former approach with *3n* qubits and a single QFT will be referred to as the *3n*x*1* approach and the latter approach with (*n*+1) qubits and QFT applied over *2n* steps as the *n*x*2n* approach.

Several earlier works have demonstrated simulations of Shor's factorization, taking advantage of the specific properties of the algorithm, towards enabling factorization of a 15-bit number (using 45 qubits) on a 16GB desktop. To the best of our knowledge, the largest number factorized so far using quantum simulations is 961307, a 20-bit number that required 37,444 CPU hours and 13.824 TB of memory on a supercomputer.

This work demonstrates simulation of Shor's factorization of the same N=961307 on a 12GB Intel Core i5 desktop in a mere 139s on a single core, using Q-Kit. Fig. 2 presents a detailed timing analysis for this simulation. It must be noted that this ~10^6x speed-up is achieved with Kitaev's *n*x*2n* approach, unlike Ref.[1]. However, even with exactly the same approach as Ref.[1], Q-Kit is still 35x faster, as shown later for a 13-bit N=8189, a 39-qubit simulation.

Furthermore, for the first time, Q-Kit was employed in simulating Shor's factorization of numbers larger than 20-bits, on a desktop. Up to 24-bit integers are successfully factorized on a 12GB Intel Core i5 machine in ~1555s (26 minutes). Shown in Fig. 3 is the result from 3 runs of Shor's factorization algorithm for 24-bit N=13564597, using the *n*x*2n* approach.

Fig. 3: Q-Kit simulations for Shor's factorization of N=13564597, a 24-bit number on a 12GB Intel Core i5 desktop. Average simulation time is 1555s. Prime factors are extracted from the measured states using continued fractions. 2 of the 3 simulation runs collapse to a measurement that can be used to successfully estimate the order from continued fractions.

The log file at the end of simulation is processed to extract a detailed timing report for different quantum gates, over all the *2n* stages of the circuit. The execution times for each gate are shown in Fig. 3 for each of the three runs. Next, the final quantum state the qubits collapse to, after measurement, is used to obtain the order of the quantum state using the method of continued fractions. As seen in Fig. 3, two of the three runs collapse to a state that can successfully estimate the order and therefore, the prime factors of N. It must be noted that although the last run was not able to estimate the prime factors, it does indeed collapse to a right quantum state (order 564840 is a multiple of 141210).

Simulations of numbers larger than N are limited by the available memory. While Q-Kit scales perfectly in memory, the simulation of the above 24-bit number already demands 8GB of memory, close to the desktop limit. 25-bit numbers need

machines with larger memory, like supercomputers or AWS.

Fig. 4 demonstrates the significantly improved performance of Q-Kit in simulating Shor's factorization of n-bit integers compared against the published data [1,2]. Q-Kit simulations are orders of magnitude faster, especially with the nx$2n$ approach, than the previously reported values. Even with the exactly same $3n$x1 approach (curve 3a), Q-Kit is ~35x faster.
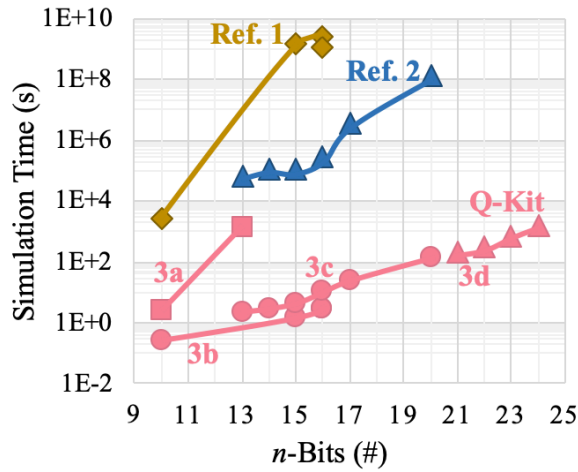

Fig. 4: Performance comparison of Q-Kit in factorizing exactly the same n-bit integers as in references [1,2]. Curves 1 and 2 correspond to Ref.[1] and Ref.[2], respectively. Curve 3a is from Q-Kit with the same $3n$x1 approach as in Ref. [1,2]. Curves 3b and 3c are also from Q-Kit for exactly the same N as in Ref.[1] and Ref.[2], respectively, but with the nx$2n$ approach. Curve 3d is from Q-Kit for factorization of 21-24 bit numbers, demonstrated here for the first time on a 12GB Intel Core i5 desktop.

Circuit.

The quantum circuit corresponding to Shor's factorization algorithm is developed using Q-Kit. Fig. 5a and Fig. 5b show the Q-Kit circuits for factorization of 4-bit N=15, based on $3n$x1 and nx$2n$, respectively. In both cases, the measurement is based on random statistical probabilities.

Simulations with $3n$x1 approach have been described in great detail in several earlier works[1,2]. The circuit uses 2*4=8 qubits in the control register and 4 qubits in the work register, without any compilation optimizations. And the quantum state of the qubits in the control register after measurement is used to extract the order. However, not every collapsed state will give the

order with the continued fractions method and a few runs might be required to land on the right state, with either of the approaches, as already shown in Fig. 3.

With the simulation based on nx$2n$ approach, a single qubit Q0 in Fig. 5b is used repeatedly 2*4=8 times and each time its measured state is saved to one of the 8 classical bits (C-1 to C-8). The classical bits are also used as a control to initialize the qubit for the next stage. The state of the 8 classical bits at the end of simulation corresponds to the final quantum state the system collapses to, as in $3n$x1 approach, and this state can be used to extract the order and prime factors using continued fractions.

Conclusion.

This work demonstrates the capabilities of Q-Kit, a graphical quantum circuit emulator. It is shown that unlike several existing quantum simulation software, Q-Kit is both user-friendly and computationally powerful. It is equipped with several useful features, including user-defined custom gates and inclusion of classical bits in quantum circuit. Q-Kit is shown to be orders of magnitude faster in simulating Shor's factorization of 20-bit numbers, and can be achieved on a desktop as against demanding resources on supercomputers. Furthermore, numbers larger than 20-bit, up to 24-bits, have been successfully simulated in less than 26 minutes for the first-time, also on a 12GB Intel Core i5 desktop.

References.

[1] De Raedt, H., Jin, F., Willsch, D., Willsch, M., Yoshioka, N., Ito, N., Yuan, S. and Michielsen, K., 2019. Massively parallel quantum computer simulator, eleven years later. *Computer Physics Communications*, *237*, pp.47-61.
[2] Dang, A., Hill, C.D. and Hollenberg, L.C., 2019. Optimising matrix product state simulations of Shor's algorithm. *Quantum*, *3*.
[3] Bennett, C.H. and DiVincenzo, D.P., 2000. Quantum information and computation. *Nature*, *404*(6775), p.247.
[4] Jozsa, R. and Linden, N., 2003. On the role of entanglement in quantum-computational speed-up. *Proceedings of the Royal Society of London.*
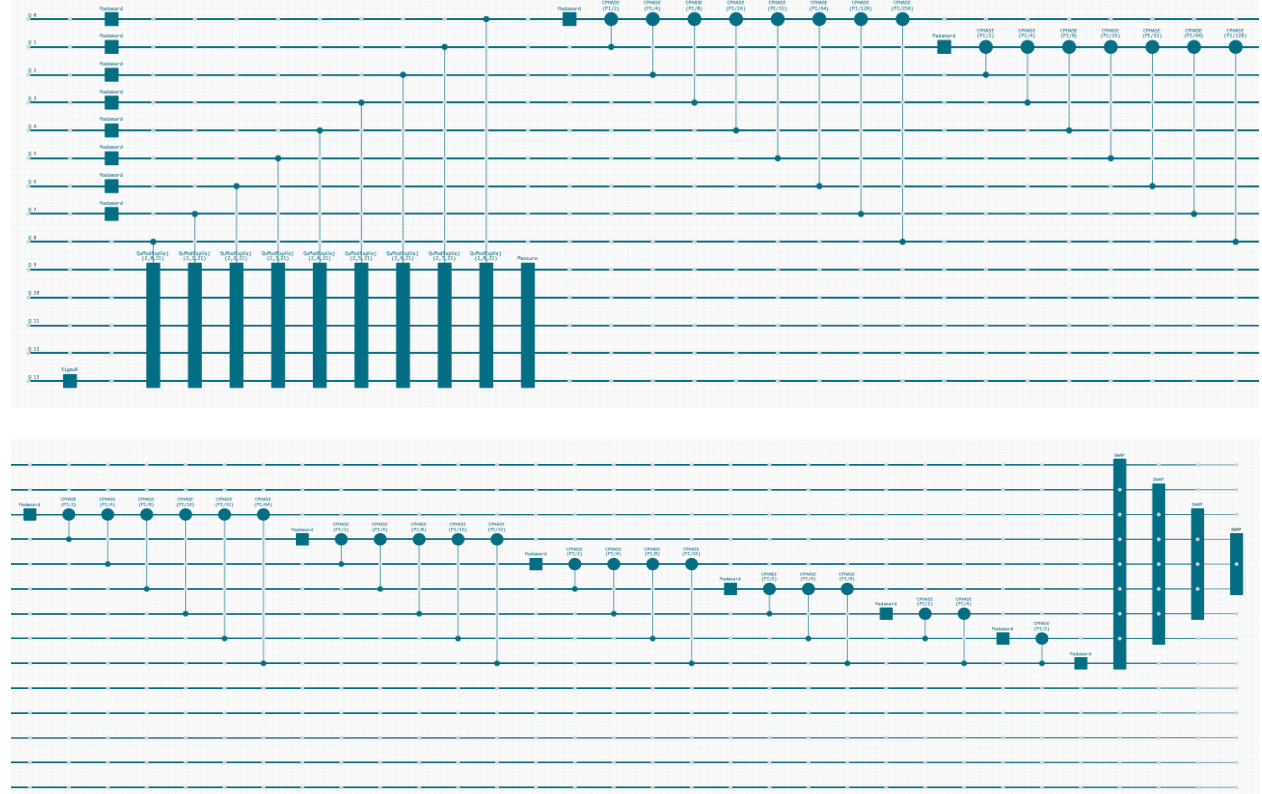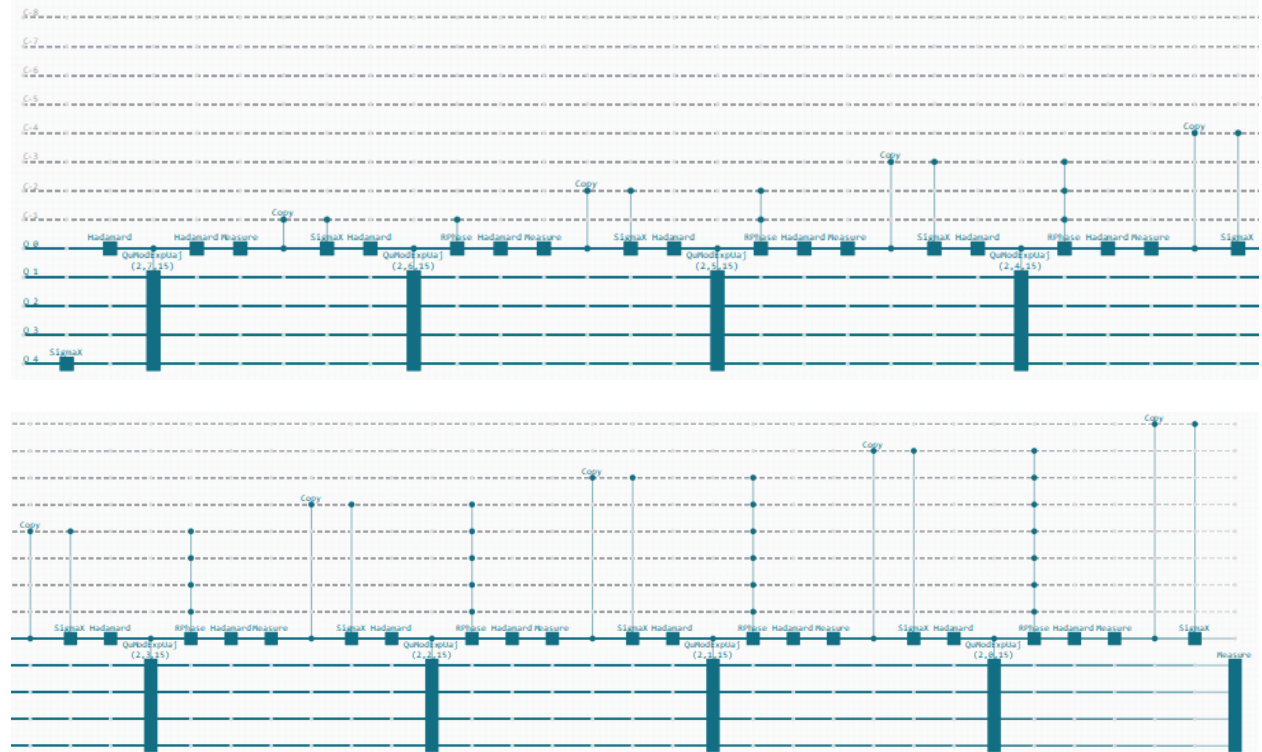
Fig. 5: Quantum circuits for Shor's factorization of 4-bit N=15. (a) Conventional full circuit with 3*4=12 qubits, 4 in work register and 8 in control register. (b) Kitaev's approach to Shor's factorization with 4 qubits in work register and a single qubit in control register, with QFT on control register over 8 stages. Each time the measurement of the control qubits is copied to a classical bit (represented by dashed line).

Series A: Mathematical, Physical and Engineering Sciences, *459*(2036), pp.2011-2032.

[5] Bouwmeester, D., Pan, J.W., Mattle, K., Eibl, M., Weinfurter, H. and Zeilinger, A., 1997. Experimental quantum teleportation. *Nature*, *390* (6660), p.575.

[6] Duan, L.M., Lukin, M.D., Cirac, J.I. and Zoller, P., 2001. Long-distance quantum communication with atomic ensembles and linear optics. *Nature*, *414*(6862), p.413.

[7] Gisin, N. and Thew, R., 2007. Quantum communication. *Nature photonics*, *1*(3), p.165.

[8] Ursin, R., Tiefenbacher, F., Schmitt-Manderbach, T., Weier, H., Scheidl, T., Lindenthal, M., Blauensteiner, B., Jennewein, T., Perdigues, J., Trojek, P. and Ömer, B., 2007. Entanglement-based quantum communication over 144 km. *Nature physics*, *3*(7), p.481.

[9] Ristè, D., da Silva, M.P., Ryan, C.A., Cross, A.W., Córcoles, A.D., Smolin, J.A., Gambetta, J.M., Chow, J.M. and Johnson, B.R., 2017. Demonstration of quantum advantage in machine learning. *npj Quantum Information*, *3*(1), p.16.

[10] Boghosian, B.M. and Taylor IV, W., 1998. Simulating quantum mechanics on a quantum computer. *Physica D: Nonlinear Phenomena*, *120*(1-2), pp.30-42.

[11] Kempe, J., 2005. Discrete quantum walks hit exponentially faster. *Probability theory and related fields*, *133*(2), pp.215-235.

[12] Bernstein, D.J., 2009. Introduction to post-quantum cryptography. In *Post-quantum cryptography* (pp. 1-14). Springer, Berlin, Heidelberg.

[13] P. W. Shor, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*(IEEE Computer Society Press, 1994), pp. 124–134.

[14] Grover, L.K., 1996. A fast quantum mechanical algorithm for database search. *arXiv preprint quant-ph/9605043*.

[15] Monz, T., Nigg, D., Martinez, E.A., Brandl, M.F., Schindler, P., Rines, R., Wang, S.X., Chuang, I.L. and Blatt, R., 2016. Realization of a scalable Shor algorithm. *Science*, *351*(6277), pp.1068-1070.

[16] Politi, A., Matthews, J.C. and O'brien, J.L., 2009. Shor's quantum factoring algorithm on a photonic chip. *Science*, *325*(5945), pp.1221-1221.

[17] Martín-López, E., Laing, A., Lawson, T., Alvarez, R., Zhou, X.Q. and O'brien, J.L., 2012. Experimental realization of Shor's quantum fact-oring algorithm using qubit recycling. *Nature Photonics*, *6*(11), p.773.

[18] Lu, C.Y., Browne, D.E., Yang, T. and Pan, J.W., 2007. Demonstration of a compiled version of Shor's quantum factoring algorithm using photonic qubits. *Physical Review Letters*, *99*(25), p.250504.

[19] Lanyon, B.P., Weinhold, T.J., Langford, N.K., Barbieri, M., James, D.F., Gilchrist, A. and White, A.G., 2007. Experimental demonstration of a compiled version of Shor's algorithm with quantum entanglement. *Physical Review Letters*, *99*(25), p.250505.

[20] Vandersypen, L.M., Steffen, M., Breyta, G., Yannoni, C.S., Sherwood, M.H. and Chuang, I.L., 2001. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, *414*(6866), p.883.

[21] De Raedt, K., Michielsen, K., De Raedt, H., Trieu, B., Arnold, G., Richter, M., Lippert, T., Watanabe, H. and Ito, N., 2007. Massively parallel quantum computer simulator. *Computer Physics Communications*, *176*(2), pp.121-136.

[22] Wang, D.S., Hill, C.D. and Hollenberg, L.C.L., 2017. Simulations of Shor's algorithm using matrix product states. *Quantum Information Processing*, *16*(7), p.176.

[23] Kitaev, A.Y., 1995. Quantum measurements and the Abelian stabilizer problem. *arXiv preprint quant-ph/9511026*.

[24] Beauregard, S., 2002. Circuit for Shor's algorithm using 2n+ 3 qubits. *arXiv preprint quant-ph/0205095*.

[25] Häner, T., Roetteler, M. and Svore, K.M., 2016. Factoring using 2n+ 2 qubits with Toffoli based modular multiplication. *arXiv preprint arXiv:1611.07995*.