양자 인수분해 알고리즘에서 리플-캐리 가산기 성능 평가

라라사티 하라스타 타티마[◆], 지 장 현[*], 박 정 환[*], 김 호 원[°]

Performance Evaluation of Ripple-Carry Adders in Quantum Factoring Algorithm

Harashta Tatimma Larasati[◆], Janghyun Ji[*], Jeonghwan Park[*], Howon Kim[°]

요 약

양자 컴퓨팅은 기술적 발전과 양자 알고리즘을 이용한 문제 해결의 확실한 장점 덕분에 관심이 증가하는 추세에 있다. 가장 잘 알려진 알고리즘 중 하나는 큰 정수를 인수분해하기 위한 쇼어 (shor) 알고리즘인데, 그 구성 요소 중 하나로 모듈러 지수 (modular exponentiation) 회로가 필요하다. 본 논문에서는 모듈러 지수 회로의 기본 구성 요소로서 효율적인 가산기(adder)를 구축하는 것의 중요성에 대해 논한다. 특히, Vedral-Barenco-Ekert (VBE), Cuccaro Adder와 같은 비교적 잘 알려진 두 개의 양자 리플-캐리 가산기 회로에 초점을 맞춘다. 또한, 본 논문에서는 회로 깊이, 폭 및 크기 측면을 Qiskit 양자 시뮬레이션을 사용하여 두 가지 시나리오로 이 회로들의 비용을 평가한다. 첫 번째로, 하나의 가산기만 사용한 경우. 두 번째로는, 쇼어 알고리즘 회로에 통합할 경우로 평가한다. 그 결과로 기본 구성 요소의 작은 개선이 쇼어 알고리즘 회로의 전체 비용을 크게 절약할 수 있다는 것을 보여준다. 마지막으로, 본 논문에서는 ETRI Q-Crypton 양자 시뮬레이터를 사용하여 Cuccaro Adder 회로의 변형 버전을 제시하고 쇼어 알고리즘에서 비용 절감 효과를 분석하여 추가적인 개선의 가능성을 탐구하는 것으로 결론을 내린다.

Key Words : quantum computing, quantum circuit, performance evaluation, ripple-carry adders, Qiskit, Q-Crypton quantum simulator

ABSTRACT

The field of quantum computation has been on the rise due to technological advances and the apparent advantages of solving problems using quantum algorithms. One of the most well-known algorithms is Shor's algorithm for factoring large integers, which requires a modular exponentiation circuit as one of its components.In this paper, we discuss the importance of building an efficient adder as the basic building block of the modular exponentiation circuit. Furthermore, we evaluate their cost in terms of circuit depth, width, and size, using Qiskit quantum simulator in two scenarios: (1) when used as a single adder, and; (2) when incorporated in Shor's algorithm circuit. The result shows that even the slightest improvement in the underlying adder translates to a relatively large saving to the overall cost of Shor's algorithm circuit. Finally, we conclude by exploring the possibilities of adder improvement by presenting a modified version of Cuccaro Adder circuit and analyzing its cost reduction in Shor's algorithm using ETRI Q-Crypton quantum simulator.

※ This work was supported by a 2-Year Research Grant of Pusan National University.
◆ First Author : Pusan National University Department of Information Convergence Engineering, harashta@pusan.ac.kr, 학생(박사과정), 학생회원
° Corresponding Author : Pusan National University Department of Information Convergence Engineering, howonkim@pusan.ac.kr, 정교수, 종신회원
* Pusan National University Department of Information Convergence Engineering, jjh0819@pusan.ac.kr, 학생(석박통합과정), 학생회원; gg2059@pusan.ac.kr, 학생(석사과정)
논문번호 : 202011-290-D-RU, Received November 20, 2020; Revised December 8, 2020; Accepted December 9, 2020.

Ⅰ. Introduction

Quantum computing has undergone a major development in the recent years. A higher number of qubits has been achieved in a shorter period of time, making the practical implementation of quantum algorithms for solving real-world problems feasible in the foreseeable future. One of the quantum algorithms with the most significant impact is Shor's algorithm, which can factor a large composite prime integer in polynomial time instead of sub-polynomial time as in the classical computation. This algorithm threatens the security of the widely-used RSA cryptosystems, which is based on the assumption of the difficulty of factoring large numbers.

One of the most computationally expensive parts of Shor's algorithm is the modular exponentiation. Among the methods that have been proposed to realize the circuit, construction from repeated addition as first proposed in [1] remains one of the most popular. In this case, an efficient addition circuit will significantly benefit the Shor's algorithm circuit in general.

Furthermore, the addition circuit is also the base of various other algorithms. Several examples are for use in quantum Toom-Cook multiplication[2], Montgomery multiplication[3], and quantum division circuit[4], which often uses addition circuit repeatedly. Hence, how to improve the existing adders will always be a relevant research question in quantum computation.

In this paper, we discuss and evaluate the performance of the quantum addition circuit. Specifically, we put our attention to two popular variants of ripple-carry adder: the naive-yet-the simplest approach adapted from the classical reversible circuit, namely the Vedral-Barenco-Ekert (VBE) adder[1]; and the Cuccaro adder[5], a slightly recent approach which has been more frequently used in the quantum algorithms. We evaluate the cost of these two adders using Qiskit quantum simulator[6] in two scenarios: for the use as a single adder; and for the use in the underlying circuit of Shor's Algorithm. Finally, we explore the improvements for further reducing the cost of ripple-carry adders, in which we take the example of the slightly modified Cuccaro adder that replaces some of its components by Peres gate and Negative-control Toffoli gate to give lower circuit depth. The evaluation using another quantum simulator that supports a more complete resource analysis, namely the ETRI Q-Crypton, shows that the modified adder gives a lower overall cost metric than the original Cuccaro adder when employed in the Shor's algorithm.

Ⅱ. Related Work

There are several works that describe the implementation of Shor's algorithm up to the gate level, such as the library provided by Qiskit[8], and a sample in Microsoft Quantum Development Kit (QDK)[9]. However, they mainly leverage the adder proposed by Draper[10] or Beauregard[11], which uses a very different concept from the ones usually found in the classical computation. Instead of utilizing binary arithmetic, these adders use a similar principle to that of the Quantum Fourier Transform (QFT), in which the computation is performed using a series of controlled rotation (cRn) gates. Even though these adders are relatively easy to instantiate in the quantum simulator and require a lower number of qubits, several argue that large circuits based on the QFT approach would not be practical in the near future due to the high number of overheads for error correction in cRn gates[12]. Therefore, current methods for constructing a modular exponentiation circuit are more into leveraging the gates that already existed in the reversible computing, such as CNOT and Toffoli gates, using binary arithmetic operations similar to the classical computation.

Regarding which options are the best fit, Rines and Chuang in [15] note that a direct cost comparison between the QFT and binary arithmetic would be difficult since each has different metrics to be considered. In this paper, we focus only on the ones utilizing binary arithmetic since they can be efficiently simulated in classical computers.

Ⅲ. Adders and Shor's Algorithm

3.1 Overview of Shor's Algorithm

Shor's algorithm[13], proposed in 1994 by Peter Shor, is one of the few examples of quantum algorithms that succeed to show a real advantage of quantum computation over the classical counterpart. In particular, the proposal shows that quantum computation can be used to solve two problems in polynomial time: (1) to decompose a large composite integer into its prime factors, and; (2) to find the discrete logarithms over finite groups[14], leveraging quantum interference to compute the period in the process. Both problems imply that in the existence of a large working quantum computer, the use of current public-key cryptography that relies on discrete logarithm problem (i.e., RSA and ECC) is deemed to be easily crackable.

Of the two variants of Shor's algorithm, the more well-studied topic is the one for factoring large numbers (also known as the quantum factoring algorithm). Comprising two registers as illustrated in Fig. 1, the algorithm starts by applying concurrent Hadamard gates to the first register, which becomes the input for the succeeding modular exponentiation circuit in the second register. The Inverse QFT (IQFT) is then applied to the first register before the result is retrieved via measurement and post-processing. Note that this algorithm is probabilistic, meaning that the correct solution may not be achieved in just one run.

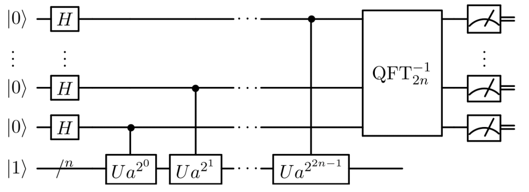


Fig. 1. General circuit for Shor's quantum factoring algorithm [7]

3.2 Adder's Role in Shor's Algorithm

In Shor's Algorithm, the most non-trivial implementation is the modular exponentiation circuit. One of the earliest explicit implementations of the circuit is presented by Vedral *et al.*, in [1], which basically proposes the use of repeated addition. Recently, more advanced modular exponentiation circuits have already been proposed, but many are essentially still leveraging the concept of repeated addition as in [1].

Additionally, an adder can also be used for performing subtraction. This is done by reversing the steps of addition, i.e., by a reversed adder circuit. Furthermore, adders are also utilized to build the circuit that performs the modular operation (i.e., operation in Galois Field). In this sense, the adder (and its reverse) can be considered as the basic building block of modular exponentiation. Since adder blocks are used extensively throughout the circuit, they are expected to be as efficient as possible.

3.3 VBE Adder

Vedral-Barenco-Ekert (VBE) adder[1] is one of the earliest and straightforward quantum reversible adders. As illustrated in Fig. 2, VBE adder consists of several types of block: the Carry block, comprising a series of Toffoli-CNOT-Toffoli gates: the Sum block, consisting of CNOT-CNOT gates; and the Reversed carry block, which undoes (i.e., uncomputes) the carry operation while also computing the sum. This uncomputation is the consequence if one wants to construct an in-place adder using these blocks, i.e., quantum operation performing $|a,b\rangle \rightarrow |a,a+b\rangle$ with a and b as the inputs.

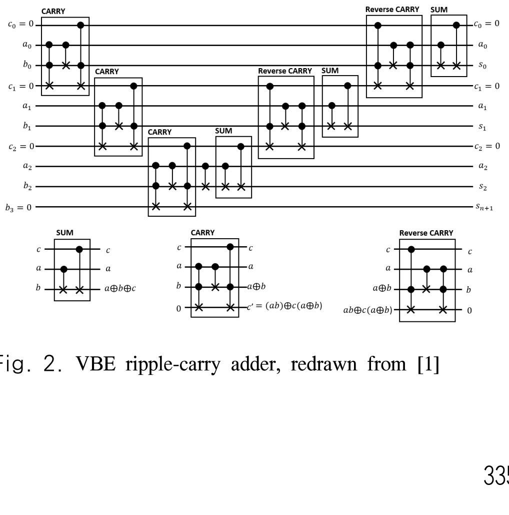


Fig. 2. VBE ripple-carry adder, redrawn from [1]

3.4 Cuccaro Adder

Cuccaro adder (sometimes also referred to as CDKM adder)[5], first proposed by Cuccaro *et al.*, is one of the first improvements in ripple-carry addition circuits. Unlike VBE adder which consists of a series of Carry and Sum gates, Cuccaro adder works by employing two different kinds of blocks: MAJ (Majority), which computes the majority of three bits, and UMA (UnMajority-and-Add), which undoes the MAJ block while also computing the sum. A series of MAJ and UMA blocks perform the addition operation, as illustrated in Fig. 3.

In [5], the authors provide two types of UMA blocks: the 2-CNOT version which is easier to understand, and the 3-CNOT version, which allows more parallelization when constructed as an adder. In the literature, Cuccaro adder and its modification variants are commonly found as the underlying circuit of proposed quantum algorithms, such as in [2,16]. This adder has a lower width than the VBE adder since it only uses one ancilla bit instead of *n-O(1).*

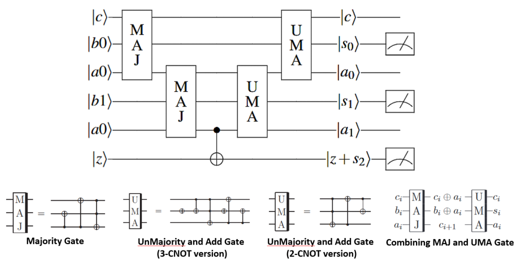


Fig. 3. Cuccaro ripple-carry adder [5]

Ⅳ. Constructing Modular Exponentiation from Addition Circuit

4.1 VBE General Modular Exponentiation Circuit

In this paper, we ground our discussion based on the modular exponentiation circuit by VBE[1] since it was the first circuit to describe in detail about the construction, while also provides the conceptually simplest circuit in the classically inspired approach. Hence, we can say that their method is the baseline for simulating and evaluating the modular arithmetic circuit. In fact, other references (e.g., [17]) have featured[1] for its detailed elaboration of modular exponentiation circuit construction for Shor's algorithm.

In [18], the authors have assembled the modular exponentiation circuit in Qiskit based on the VBE approach. Specifically, they simulated the 4-bit VBE modular exponentiation of modulus value N=15. However, they did not go further to the valid Shor's algorithm for simulation (i.e., run the circuit until performing measurement, and employing 8 bit instead of only 4 bit in the argument register) due to the limited number of QASM instructions. Nevertheless, the resource estimates and the figures are still able to be acquired. In this section, we provide their circuit for Shor's algorithm briefly to ease the comprehension of reader on the importance of adder as well as for completeness.

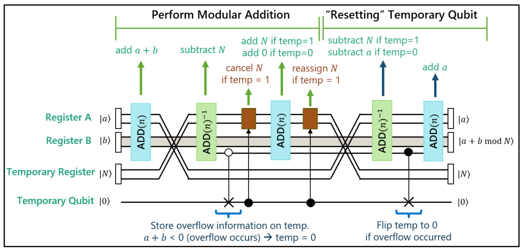
Essentially, VBE[1] proposes these steps to realize the modular exponentiation circuit as follows.

1. Adder, which outputs $a+b$, $0 \leq a,b$;
2. Modular adder, which outputs $a+b$ mod *N*;
3. Modular multiplexer, which outputs ax mod *N*;
4. Modular multiplier, which outputs a^x mod *N*.

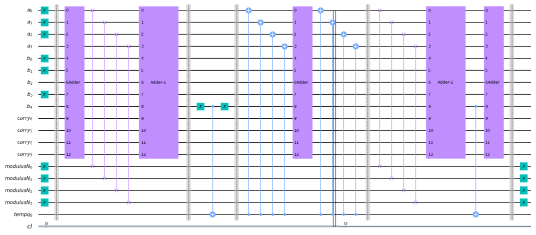
For Shor's quantum factoring algorithm, the exponentiation is by a constant, meaning that we only have one variable as input while the other value is hardwired. In the notation, a represents the constant value, x points to the variable input, and *N* denotes the modulus value to be factored and is always fixed during the whole computation. Note that the a and x in point 3 and 4 above do not necessarily represent the same values, whereas *N* always refers to the same value during computation.

4.2 Modular Adder

Modular adder can be built by five adders, two of which are reversed version that performs subtraction. It outputs $a+b-N$ when the result is still greater than 0, otherwise it will not perform subtraction by *N* (i.e., outputs $a+b$ instead). As illustrated in Fig. 4 (taken from [18]), three adders are required for the modular addition while the other two are only for restoring the temporary qubit back to $|0\rangle$.



(a)



(b)

Fig. 4. Modular adder circuit: (a) redrawn from [1][15], (b) presented in [18].

4.3 Modular Multiplier

Modular multiplier circuit is constructed from a series of modular adders by utilizing the binary expansion of x for the constant and making a as hardwired, as presented in Eq. (1).

$$x = ax_{n-1}2^{n-1} + ... + ax_1 2^1 + ax_0 2^0 = \sum_{x_k} a2^k \qquad (1)$$

The circuit is as illustrated in Fig. 5. Note that the required circuit is the controlled version rather than the general one, with the state mapping is as defined in [17], presented in Eq. (2).

$$\mathrm{CMODMULT}(n)|c,x,0,0\rangle = \begin{cases} |c,x,0,ax \bmod N\rangle & (c=1) \\ |c,x,0,x\rangle & (c=0) \end{cases} \qquad (2)$$

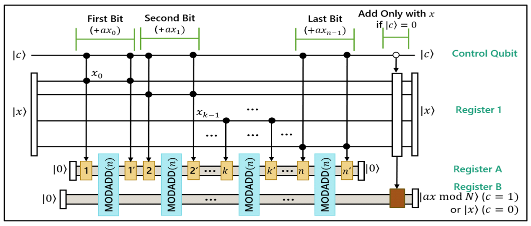
Additionally, after a number is assigned and modular addition is performed, the numbers to be appended (e.g., $a.2^0$) are uncomputed, to be replaced by the consequent numbers (e.g., $a.2^1$). Note that in the figure, the first part to be computed is the lowest significant bit (LSB).

4.4 Modular Exponentiation

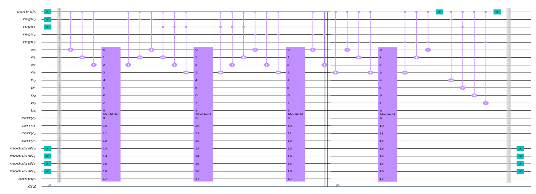
In terms of modular exponentiation, the circuit realizes Eq. (4) and the mapping is $|x,1,0\rangle = |x,a^x \mathrm{mod}\ N,0\rangle$.

$$a^x = a^{x_{n-1}2^{n-1}} \times a^{x_{n-2}2^{n-2}} \times \cdots a^{x_1 2} \times a^{x_0} = \prod_{\substack{k=0 \\ x_k=1}}^{n-1} a^{2^k} \qquad (4)$$

However, compared to the previous, this circuit is not as straightforward to build. It requires swapping the values and appending the multiplier that computes the inverse of the preceding circuit, as

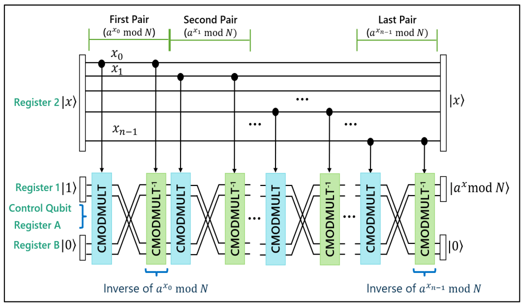


(a)

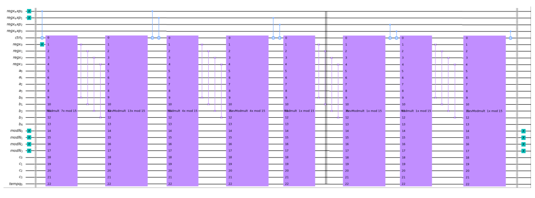


(b)

Fig. 5. Modular multiplier circuit: (a) redrawn from [1][15], (b) presented in [18].


First Pair (a^{x_0} mod N)
Second Pair (a^{x_1} mod N)
Last Pair ($a^{x_{n-1}}$ mod N)
Register 2 $|x\rangle$
Register 1 $|1\rangle$
Control Qubit
Register A
Register B $|0\rangle$
CMODMULT
CMODMULT^{-1}
$|a^x \bmod N\rangle$
Inverse of a^{x_0} mod N
Inverse of $a^{x_{n-1}}$ mod N

(a)



(b)

Fig. 6. Modular exponentiation circuit: (a) redrawn from [1][15], (b) presented in [18].

shown in Fig. 6. For a more detailed explanation, interested readers may refer to [18].

Ⅴ. Evaluation of Ripple-Carry Adders

5.1 Experiment Method and Setup

In this section, we perform several experiments in IBM Qiskit quantum simulator to evaluate the performance of VBE and Cuccaro adder. Even though a mathematical formulation is possible, in this study, we leverage the result obtained by the quantum simulator to acquire a closer behavior to quantum hardware. Currently, the performance metrics available in Qiskit are circuit width, depth, and size. Circuit width corresponds to the number of qubits and classical bits; circuit depth represents the length of critical path or the lowest number of time steps; circuit size refers to the number of gates in the circuit.

For the first scenario, we simulate the adders as a standalone (i.e., evaluate a single adder only, without employing it into a larger circuit) and evaluate them for an increasing number of inputs. In particular, we obtain the resource count for the bit length from 4-bit up to 300-bit addition circuit, with zeros as the input of the adders. We provide the result in two types of compilation level supported by Qiskit: (1) the standard version, which compiles as-is (i.e., only to the logical layer); and (2) the decomposed version, which expands to a lower-level compilation.

For the second scenario, we evaluate each adder when incorporated as the building block of Shor's algorithm to investigate how the resource requirement scale for larger circuits. In this case, we only run from 4-bit up to 16 bitlength of the number to be factored (i.e., modulus value *N*) due to our limited resource. Additionally, the measured metrics for this scenario are circuit width, depth (decomposed), and size (decomposed).

Instead of zeroing the inputs as in the first scenario, in the second scenario, we run the circuit with a valid value of a and N accordingly (i.e., 7^x mod 15 for 3-bit factoring, 5^x mod 21 for 4-bit factoring, and so on). Additionally, to closely follow the requirement of Shor's algorithm (which is often overlooked in the existing simulations), we prepare the first register in size $N^2 \leqq 2^{size} < 2N^2$ qubits.

In terms of the experiment environment, we run the evaluation on Qiskit version 0.14.0 in our local PC, with the specification of Windows 10 Pro, Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, 6 Cores, 12 logical processors, and 64 GB of RAM. Regarding the code simulation of Shor's algorithm in Qiskit, we refer to the scheme in [18].

Note that in the next section (Section VI), we perform one additional evaluation, that is to compare between Cuccaro adder and its modified version, which is expected to improve the original circuit and give a lower cost. Different from this section, the experiments in the next section are performed in ETRI's Q-Crypton, a web-based quantum simulator that provides not only a simulation but also an in-depth resource analysis of a quantum circuit.

5.2 Evaluation of Single VBE and Cuccaro Adder

For comparing the cost of standalone VBE and Cuccaro adder, the result of the experiment is presented in Fig. 7, with *x*-axis representing the *n*-bit addition and *y*-axis depicting all of the cost metrics in logarithmic scale. The dashed and solid lines in the figures belong to VBE and Cuccaro adder, respectively.

Overall, Cuccaro adder yields lower cost in almost all of the metrics except one. In particular, for the same *n*-bit addition, metrics with the highest

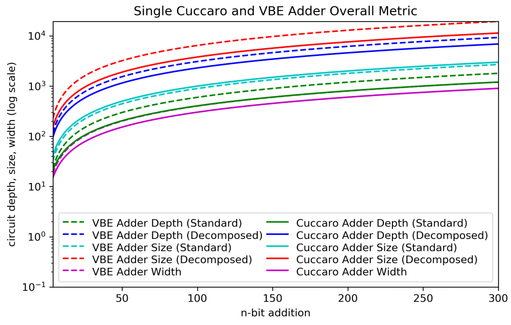


Fig. 7. Cost comparison of single VBE and Cuccaro adder

cost reduction is in the following order: circuit size (decomposed), depth (standard), depth (decomposed), and width. Furthermore, as the bit increases, the cost reduction percentage also shows a trend of increase. In other words, the efficiency is more apparent for larger circuits, as presented in Table 1.

For instance, for a 5-bit addition, the improvements in terms of width, depth (standard), depth (decomposed), and size (decomposed) are around 18, 14, 19, and 35 percent, respectively, whereas for a 6-bit addition, the number scales to around 21, 24, 22, and 38 percent, respectively. Note that above 100-bit addition, the percentage of increase becomes more gentle; for 300-bit addition, the number stays around 25, 33, 26, and 41 percent, respectively.

Nevertheless, in terms of size (standard), VBE adder shows a lower cost. This is because we use the 3-CNOT version of UMA gate in Cuccaro adder to achieve more parallelization. Additionally, readers may notice that the performance difference in the standard compilation and decomposed version seems contradictory. However, there is an explanation for this occurrence; that is because VBE employs more Toffoli gates than Cuccaro adder. In a decomposed version, the compiler unrolls the instruction to a lower level. While CNOT gates are native to superconducting circuits as the base architecture of Qiskit quantum simulator, each Toffoli gate is dissected into a series of Hadamard and T gates, resulting in a higher gate count in the real implementation.

Table 1. Percentage of Improvement of Cuccaro Adder over VBE Adder for Selected Values of *n*

Improvement of Cuccaro Adder over VBE Adder (%)					
n	Width	Depth (Standard)	Depth (Decomposed)	Size (Standard)	Size (Decomposed)
5	18.18%	13.79%	18.92%	-24.39%	34.81%
10	21.43%	23.73%	22.44%	-17.44%	38.35%
50	24.26%	31.44%	25.15%	-12.33%	40.93%
100	24.63%	32.39%	25.48%	-11.72%	41.23%
150	24.75%	32.70%	25.59%	-11.52%	41.34%
200	24.81%	32.86%	25.64%	-11.41%	41.39%
250	24.85%	32.96%	25.67%	-11.35%	41.42%
300	24.88%	33.02%	25.70%	-11.31%	41.44%

5.3 Evaluation of VBE and Cuccaro Adder in Shor's Algorithm

Regarding the overall cost in Shor's algorithm, we separate the figures for different metrics for ease of reading. Additionally, we do not employ a logarithmic scale since the *x*-axis only covers a small value (only up to 16-bit modulus). The comparison of depth, width, and size of Shor's algorithm utilizing VBE and Cuccaro adder is shown in Fig. 8, Fig. 9, and Fig. 10, respectively, and the percentage of overall performance is presented in Table 2. Note that for this subsection, width and size refer to the decomposed version rather than the standard version.

As shown in the figures, different adders yield a considerable cost difference in Shor's algorithm, even though we only use small bitlengths for our experiment. In terms of circuit depth and width, Cuccaro adder gives lower cost, which is consistent

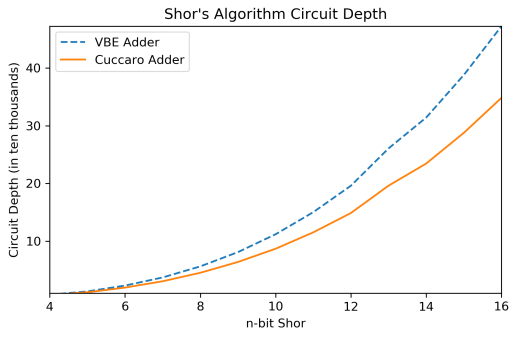


Fig. 8. Cost comparison of single VBE and Cuccaro adder

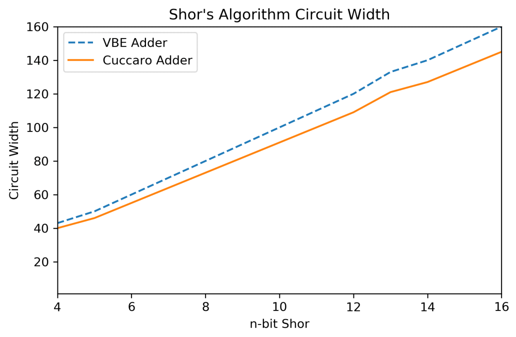


Fig. 9. Cost comparison of single VBE and Cuccaro adder

Table 2. Improvement of Shor's Algorithm using Cuccaro Adder over VBE Adder, from n=3 to n=16

n	Modulus (N)	First Register Size	Improvement of Shor's Algorithm using Cuccaro Adder over VBE Adder (%)		
			Width	Depth (Decomposed)	Size (Decomposed)
4	15	8	6.98%	6.66%	-26.69%
5	21	9	8.00%	11.54%	-23.97%
6	33	11	8.33%	15.11%	-21.99%
7	87	13	8.57%	17.79%	-20.08%
8	177	15	8.75%	19.66%	-19.11%
9	291	17	8.89%	21.06%	-18.26%
10	537	19	9.00%	22.41%	-17.63%
11	1041	21	9.09%	23.28%	-17.12%
12	2049	23	9.17%	24.13%	-16.74%
13	6393	26	9.02%	24.75%	-15.98%
14	8193	27	9.29%	25.36%	-16.01%
15	18339	29	9.33%	25.84%	-15.42%
16	34761	31	9.38%	26.26%	-15.18%

with what we have previously seen in the single adder implementation.

As presented in Fig. 8, the difference in depth is considerably high. For instance, in the 16-bit case, the depth of Shor's algorithm employing VBE adder is around 472 thousand while it is only 348 thousand when Cuccaro adder is used, or around 26 percent. Additionally, as shown in Fig. 9, an improvement of around 9 percent (145 as opposed to 160) is achieved in terms of width. Even though the level of improvement is not as high as that of the depth, circuit width is also a scarce resource because it translates to the required number of logical qubits to run a quantum algorithm, and its reduction is not as easy. Hence, 9 percent of improvement can be considered as significant.

However, in terms of circuit size, it is the VBE adder that yields a lower cost, as shown in Fig. 9. The reason why the result is different from that of the standalone adder in the previous subsection can be because the decomposition is only into one layer below. Since the adder is incorporated into a larger circuit (i.e., modular adder in our implementation), the circuit's unrolling may still follow the adder's standard compilation.

Nevertheless, inferring from Table 2, as the circuit enlarges, the results seem to be in favor of the Cuccaro adder. Not only the circuit efficiency in terms of width and depth increases, but also the size

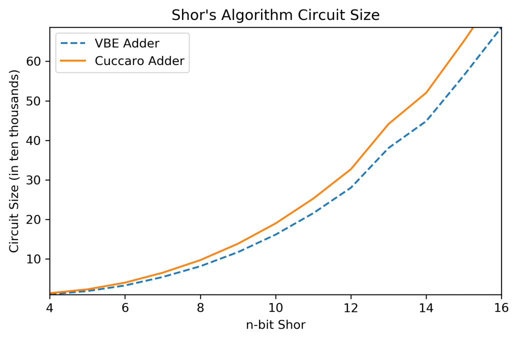


Fig. 10. Circuit size comparison of Shor's algorithm employing VBE and Cuccaro adder

gap becomes lower. For the 4-bit Shor, the cost improvement in width, depth, and size is around 7, 7, and −26 percent, respectively, while for the 16-bit, it leaps to around 9, 26, and −15, respectively. Hence, in Shor's algorithm's real use (e.g., to crack RSA-2048), a much larger cost reduction can be acquired by equipping the suitable addition circuit.

Ⅵ. Possibility of Improvements

6.1 Improvements in Literature

To reduce the cost −or to improve the performance− of an adder, one may experiment and analyze from the existing quantum addition circuit. In the literature, there have also been various proposals that aim to reduce the integer addition cost in one (or more) aspects of performance metrics. One example is the adder by Takahashi *et al.*[19], also known as the TTK adder, which reduces the qubit size and, to date, is known as the adder with the lowest width. Gidney *et al.*[16], proposes a modification to Cuccaro adder with the primary aim to reduce the number of T gates in one stage (also called the "T-depth"), which mainly comes from the number of Toffoli gates in the circuit. This proposal, also referred to as the CDKMG adder, utilizes $4n$ T gates, but with the tradeoff of n auxiliary qubits. Additionally, a paper by Draper *et al.*[20], targets on reducing the T-depth, giving a circuit with logarithmic depth. The reason why many proposals focus on reducing the depth is because it describes the time complexity of a circuit[21], which is essential to be minimized since the decoherence time in the current quantum hardware is still relatively small.

Another approach to cost reduction is by considering another gate model as proposed in [22]. The paper discusses the possibility of utilizing Peres gate, which is relatively well known earlier in the reversible computing field and has the lowest quantum cost when decomposed to NCV library (i.e., the set that comprises {NOT, CNOT, V, V† } gates). When Clifford+T, i.e., the set of gates comprises {NOT, CNOT, H, Z, S, S† , T, T† } becomes a more preferred implementation for a fault-tolerant library, the discussion of Peres gate had subsided, but emerges again when Amy *et al.*, in [23] presents the decomposition of Peres gate to Clifford+T library. The operation of Peres gate is illustrated in Fig. 11.

In [22], the possibility of constructing the Cuccaro adder based on the gates in Fig. 11 is discussed. Specifically, the aim is to reduce the circuit's depth by replacing a particular part of the Cuccaro adder by Peres gate and negative-control Toffoli gate as illustrated in Fig. 12. Essentially, a consecutive Toffoli-CNOT gate is replaced by the decomposition of Peres gate to Clifford+T library, while the commuting X-Toffoli-X gate is replaced by the decomposition of negative-control Toffoli as shown in Fig. 12.

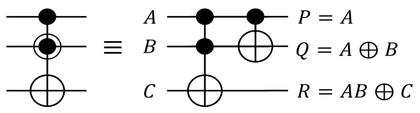


Fig. 11. Operation of Peres Gate

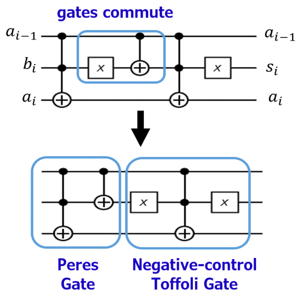


Fig. 12. Modification of Cuccaro adder to reduce depth in [22]

For the work in [22], a simulation is done in ETRI Q-Crypton quantum simulator. This simulator supports a resource estimation, namely the "DNA Analysis", that provides a relatively more complete list of performance metrics, ranging from algorithm level (namely, "Compile level") to the fault-tolerant quantum computation ("FTQC level"). In [22], the authors show the performance comparison for 3-bit addition, which yields improvements ranging from 3 to 40 percent. As for larger qubits, the simulation and resource estimation are run for 3 to 10-bit addition, which also shows a considerable cost saving.

6.2 Cost Reduction of Modified Cuccaro Adder in Q-Crypton Quantum Simulator

We are inspired by the work of [22] to continue investigating the cost reduction of their modified Cuccaro adder with the original one by incorporating each of them into the Shor's algorithm circuit. For this experiment, instead of using Qiskit like the experiment in the previous section, we instead utilize Q-crypton quantum simulator to acquire a more complete cost estimation.

The script of Shor's algorithm is written by us, which yields the FTQC level's cost as presented in Table 3 and Table 4 for the original Cuccaro and the modified version, respectively. Note that the

Table 3. Performance metric of Cuccaro adder in Shor's algorithm, from *n*=3 to *n*=10

Cuccaro Adder (Original)							
Bits to factor	**4**	**5**	**6**	**7**	**8**	**9**	**10**
FTQC Level							
Algorithm Qubits	24	29	34	39	44	49	54
Circuit Depth	14,114	27,118	46,078	72,422	107,206	152,158	207,626
Computing Time	54.17	104.25	177.32	278.90	413.10	586.59	800.73
KQ	338,736	786,422	1,566,652	2,824,458	4,717,064	7,455,742	11,211,804
Logical Gates	27,675	52,805	89,900	141,300	209,345	296,375	404,730
Physical Qubits	24	29	34	39	44	49	54

Table 4. Performance metric of modified Cuccaro adder in Shor's algorithm, from *n*=3 to *n*=10

Cuccaro Adder (Improved Version)							
Bits to factor	**4**	**5**	**6**	**7**	**8**	**9**	**10**
FTQC Level							
Algorithm Qubits	24	29	34	39	44	49	54
Circuit Depth	13,474	25,868	43,938	69,012	102,086	144,868	197,646
Computing Time	51.32	98.34	166.80	261.71	386.83	548.65	748.18
KQ	323,376	750,172	1,493,892	2,691,468	4,491,784	7,098,532	10,672,884
Logical Gates	26,235	49,805	84,500	132,480	195,905	276,935	377,730
Physical Qubits	24	29	34	39	44	49	54

result for Compile level is omitted for brevity since it is also reflected on the FTQC level. Furthermore, there may be differences between resource estimates in Q-Crypton and Qiskit due to different settings, compilation, and decomposition methods between the two simulators.

As shown in Table 5, the modified Cuccaro adder gives a cost reduction over the original one, ranging from 4 to 6 percent when employed in Shor's algorithm. Even though the difference is less evident when discussed in percentage compared to between VBE and Cuccaro adder, from Table 3 and Table 4, the difference in values is quite large, such as in the circuit depth. Hence, the use of an improved adder may indeed yield a significant cost saving for Shor's algorithm.

Table 5. Improvement of the modified compared to original Cuccaro adder in Shor's algorithm, from *n*=3 to *n*=10

Performance Increase (Improvements over Standard Version)							
Bits to factor	**4**	**5**	**6**	**7**	**8**	**9**	**10**
FTQC Level							
Algorithm Qubits	-	-	-	-	-	-	-
Circuit Depth	4.53%	4.61%	4.64%	4.71%	4.78%	4.79%	4.81%
Computing Time	5.26%	5.67%	5.93%	6.16%	6.36%	6.47%	6.56%
KQ	4.53%	4.61%	4.64%	4.71%	4.78%	4.79%	4.81%
Logical Gates	5.20%	5.68%	6.01%	6.24%	6.42%	6.56%	6.67%
Physical Qubits	-	-	-	-	-	-	-

Ⅶ. Conclusion

In this paper, we have discussed two well-known variants of quantum ripple-carry adders, namely the VBE and Cuccaro adder. We started by elaborating on the roles of adders and their part in the construction of larger circuit, with modular exponentiation for Shor's algorithm as the example. From our experiment in Qiskit, it has been shown that Cuccaro adder outperforms the VBE adder for standalone use for almost all metric, whereas for the use in Shor's algorithm, it excels than VBE in terms of circuit depth and width, but with a tradeoff of a larger circuit size for our current implementation. However, it should be noted that there is no one-size-fits-all option; choosing which adder to use in general will heavily depend on the predetermined cost metrics that one wants to achieve. Thus, we also discussed the recent proposals in the quantum adders to reduce some specific performance metrics, along with the example of improvement by utilizing Peres gate and negative-control Toffoli gate for use in Shor's algorithm, evaluated using Q-Crypton simulator. To conclude, the seemingly small improvement in adders yields a considerable cost reduction for Shor's algorithm circuit and, thus, shall always be pursued.

References

[1] V. Vedral, A. Barenco, and A. Ekert, "Quantum networks for elementary arithmetic operations," *Phys. Rev. A - At. Mol. Opt. Phys.*, vol. 54, no. 1, pp. 147-153, 1996. doi: 10.1103/PhysRevA.54.147.

[2] S. Dutta, D. Bhattacharjee, and A. Chattopadhyay, "Quantum circuits for Toom-Cook multiplication," *Phys. Rev. A*, 2018. doi: 10.1103/PhysRevA.98.012311.

[3] M. Roetteler, M. Naehrig, K. M. Svore, and K. Lauter, "Quantum resource estimates for computing elliptic curve discrete logarithms," *LNCS (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10625, pp. 241-270, 2017. doi: 10.1007/978-3-319-70697-9_9.

[4] H. Thapliyal, E. Munoz-Coreas, T. S. S. Varun, and T. Humble, "Quantum circuit designs of integer division optimizing T-count and T-depth," *IEEE Trans. Emerg. Top. Comput.*, 2019. doi: 10.1109/TETC.2019.2910870.

[5] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "*A new quantum ripple-carry addition circuit*," pp. 1-9, 2004. [Online]. Available: http://arxiv.org/abs/quant-ph/0410184.

[6] H. Abraham, et al., "*Qiskit: An open-source framework for quantum computing*," Qiskit research community, Jan. 2019. doi: 10.5281/zenodo.2562110.

[7] Bender2k14, "Quantum subroutine in Shor's algorithm," *Quantum*, vol. 1, no. 2000, pp. 7-10, 2006.

[8] Qiskit Development Team, "*Shor's algorithm*," Retrieved Jul. 26, 2020 from https://qiskit.org/

textbook/ch-algorithms/shor.html.

[9] T. Matsuzaki, "*Programming quantum arithmetics (Adder, Multiplier, and Exponentiation)-tsmatz*," Retrieved Jul. 26, 202, from https://tsmatz.word press.com/2019/05/22/quantum-computing-modulus-add-subtract-multiply-exponent/.

[10] T. G. Draper, "*Addition on a Quantum Computer*," pp. 1-8, 2000. [Online]. Available: http://arxiv.org/abs/quant-ph/0008033.

[11] S. Beauregard, "Circuit for Shor's algorithm using 2n+3 qubits," *Quantum Inf. Comput.*, vol. 3, no. 2, pp. 175-185, 2003.

[12] T. Häner, M. Roetteler, and K. M. Svore, "Factoring using 2n + 2 qubits with toffoli based modular multiplication," *Quantum Inf. Comput.*, vol. 17, no. 7-8, pp. 673-684, 2017.

[13] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484-1509, 1997. doi: 10.1137/S0097539795293172.

[14] J. Proos and C. Zalka, "*Shor's discrete logarithm quantum algorithm for elliptic curves*," 2003. [Online]. Available: http://arxiv.org/abs/quant-ph/0301141.

[15] R. Rines and I. Chuang, "*High performance quantum modular multipliers*," pp. 1-48, 2018. [Online]. Available: http://arxiv.org/abs/1801.01081.

[16] C. Gidney and M. Ekerå, "*How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits*," pp. 1-25, 2019. [Online]. Available: http://arxiv.org/abs/1905.09749.

[17] M. Nakahara and T. Ohmi, *Quantum computing: From linear algebra to physical realizations*, CRC press, 2008.

[18] H. T. Larasati and H. Kim, "Simulation of modular exponentiation circuit for shor's algorithm in qiskit," in *Proc. 14th Int. Conf. TSSA*, pp. 1-7, Bandung, Indonesia, 2020. doi: 10.1109/TSSA51342.2020.9310794.

[19] Y. Takahashi, S. Tani, and N. Kunihiro, "Quantum addition circuits and unbounded fan-out," *Quantum Inf. Comput.,* vol. 10, no. 9-10, pp. 872-890, 2010.

[20] T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore, "*A logarithmic-depth quantum carry-lookahead adder*," pp. 1-21, 2004. [Online]. Available: http://arxiv.org/abs/quant-ph/0406142.

[21] L. Gyongyosi and S. Imre, "Circuit depth reduction for gate-model quantum computers," *Sci. Rep.,* 2020. doi: 10.1038/s41598-020-67014-5.

[22] H. T. Larasati, J. Ji, and H. Kim, "An improvement of quantum ripple-carry addition circuit," in *Proc. CISC-W*, Korea, 2020.

[23] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 32, no. 6, pp. 818-830, 2013. doi: 10.1109/TCAD.2013.2244643.

라라사티 하라스타 타티마

(Harashta Tatimma Larasati)

2016년 : Institut Teknologi Bandung (Indonesia) Telecomunication Engineering (학사)

2017년 : Institut Teknologi Bandung (ITB) Electrical Engineering (석사)

2019년~현재 : 부산대학교 정보융합공학과 (박사과정)

<관심분야> 컴퓨터 네트워크, 양자컴퓨팅

[ORCID:0000-0001-6143-4134]

지 장 현 (Janghyun Ji)

2016년 2월 : 부산대학교 정보컴퓨터공학부 (학사)

2016년 3월~현재 : 부산대학교 전기전자컴퓨터공학부 (석박통합과정)

<관심분야> 암호 및 보안, 하드웨어 보안, 양자컴퓨팅

[ORCID:0000-0001-7299-9827]

박 정 환 (Jeonghwan Park)

2020년 8월 : 부산대학교 전기컴퓨터공학부 (학사)

2020년 9월~현재 : 부산대학교 정보융합공학과 (석사과정)

<관심분야> 암호 및 보안, 네트워크 보안, 핀테크

김 호 원 (Howon Kim)

1993년 : 경북대학교 전자공학과 졸업 (학사)

1995년 : 포항공과대학교 전자전기공학과 (석사)

1999년 : 포항공과대학교 전자전기공학과 (박사)

1998~2008년 : 한국전자통신연구원 (선임연구원/팀장)

2008년~현재 : 부산대학교정보컴퓨터공학부교수

2014년~현재 : 부산대학교사물인터넷연구센터센터장

2020년~현재 : 부산대학교 지능형융합보안대학원 책임교수

2020년~현재 : 부산대학교 블록체인 플랫폼 연구센터 센터장

<관심분야> 인공지능 및 지능형 IoT, 암호 및 보안, 블록체인

[ORCID:0000-0001-8475-7294]