Optimized Quantum Circuit Implementation of Payoff Function

Sejin Lim; Hansung University, Seoul, Korea; Email: dlatpwls834@gmail.com
Hyunjun Kim; Hansung University, Seoul, Korea; Email: khj930704@gmail.com
Kyungbae Jang; Hansung University, Seoul, Korea; Email: starj1023@gmail.com
Siyi Wang; Nanyang Technological University, Singapore; Email: siyi002@e.ntu.edu.sg
Anubhab Baksi; Nanyang Technological University, Singapore; Email: anubhab.baksi@ntu.edu.sg
Anupam Chattopadhyay; Nanyang Technological University, Singapore; Email: anupam@ntu.edu.sg
Hwajeong Seo; Hansung University, Seoul, Korea; Email: hwajeong84@gmail.com

***Abstract*—Large-scale quantum computers that can execute practical quantum algorithms have the potential to solve complex problems that are currently challenging for classical computers. This involves converting these problems into a form that can be processed by quantum circuits, a crucial process that requires minimizing quantum resources like qubit count, gate count, and circuit depth. Our work focuses on implementing and optimizing the foundational task of quantum finance, known as option pricing, as a quantum circuit. This enables the utilization of quantum computing benefits, within the financial domain. Specifically, we implement and optimize the function $f_K(S) = \max(S - K, 0)$. Taking into consideration the significant trade-offs between qubit count and circuit depth, we have developed quantum circuits for the optimized implementation of the $f_K(S)$. Our work incorporates various optimization techniques for the circuit, such as selecting the optimal adder, optimizing the $S - K$ operation, parallelization, and qubit reuse. Furthermore, we offer various versions of our quantum circuits for the $f_K(S)$, each featuring different adders and Toffoli decompositions, thereby providing flexibility for a wide range of use cases.**

***Index Terms*—Quantum circuit, Finance arithmetic, Call option payoff function**

I. Introduction

Quantum computers, leveraging unique computational paradigms distinct from classical computers, are rapidly gaining traction across numerous industries. This surge in interest can be attributed to their superior ability to solve complex tasks that are computationally challenging for classical machines. Quantum computers achieve this by employing quantum phenomena like superposition and entanglement to process information more efficiently, leading to notable reductions in computation time and significant improvements in solution accuracy.

One of the primary sectors where quantum computing's potential is being actively explored is the Financial Services Sector (FSS). Over the past few years, the financial industry has seen an increasing volume of research focused on harnessing the power of quantum computing [1], [2], [3], [4]. This interest stems from the financial sector's role in managing and operating vast amounts of capital, where minute computational improvements can yield substantial returns or risk mitigations.

By creating quantum algorithms tailored for portfolio optimization, risk analysis, and pricing of financial derivatives, the industry could revolutionize its operations and drive significant growth and profitability. The quantum-era financial models could potentially address computational bottlenecks currently faced by classical systems, enabling faster, more accurate financial forecasting and decision-making.

In this context, our paper presents a quantum circuit optimized for computing the quantum financial arithmetic function $f_K(S) = max(S - K, 0)$, commonly referred to as *call option payoff* function. This function holds significant importance within the financial market, serving as a pivotal tool for devising investment strategies and managing risks associated with fluctuations in stock prices and underlying assets. We refer to $f_K(S)$ as call option payoff function or payoff throughout this paper. Our proposed circuit can potentially enable faster and more efficient computation of this call option payoff function on quantum hardware, enhancing the overall efficiency of financial models that use them.

Within the framework of physical quantum computers, the presence of noise errors presents a significant challenge to their reliable and accurate operation [5]. To address this issue, fault-tolerant operations are utilized with quantum error correction codes based on Clifford+ T gates [6], [7]. However, implementing fault-tolerant operations incurs overhead [6], [8], [9], especially with T gate implementation. Therefore, recent research focuses on evaluating the cost of quantum circuits using metrics such as qubit count, T-count, and T-depth [5], [7], [10]. In our understanding, the most critical trade-offs in quantum circuit implementation arise from qubit count and circuit depth (gate count is somewhat independent and not subject to this trade-off).

In this regard, we delve into a comparative analysis of quantum arithmetic for call option payoff function, with a specific focus on performance metrics such as qubit count, T-count, and T-depth. Our analysis includes various Toffoli decompositions applied to different quantum adder circuits, such as the ripple carry adder and the carry look-ahead adder.

Our Contribution

1) ***Call option payoff* function for Quantum Computers**. We present the optimized quantum circuit implementation of call option payoff function (see Section II-A for its

979-8-3503-2599-7/23/$31.00 ©2023 IEEE

description), which is one of the essential components in quantum finance. To the best of our knowledge, this is the first implementation of call option payoff function for quantum computers.

2) **Implementations/Methodologies for Reducing Qubit Count and Depth**. We optimize two metrics, qubit count and depth, in the implementation of the quantum circuit for call option payoff function. Specifically, *space-efficient* implementations that reduce the qubit count and *time-efficient* implementations that minimize the depth are presented in this work.
3) **A Comprehensive Comparison of Various Adders and Toffoli Decompositions Applied**. We implement various versions of call option payoff function quantum circuits using a variety of quantum adders (such as, ripple carry adder and carry look-ahead adder). Furthermore, we estimate the quantum resources in detail at the Clifford + T level by replacing the different Toffoli gate implementations/decompositions. We gather multiple implementations to compare performance and enable the selection of the optimal call option payoff function for a given quantum computing environment.

II. Background

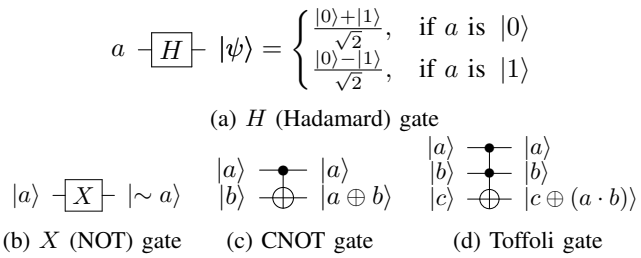
A. Call Option Payoff Function

A *call option* is the right to buy an asset such as a stock, currency, or commodity at a predetermined price (strike price) until a specific date (expiration). The right to exercise this option is optional, and if the price of the asset exceeds the strike price, the call option holder can buy the asset at the strike price and make a profit. If the price falls, the call option holder may choose not to exercise the option and instead purchase the asset on the market, which may be more advantageous.

The *option payoff* function of a *call option*, namely the $max(S-K,0)$ function, is used to calculate the profit or loss that results from the call option contract, taking into account the price (S) of the underlying asset and the strike price (K). This function is used to calculate the profit of a call option contract, and its value is determined by subtracting the strike price (K) from the price (S) of the underlying asset, resulting in the value of $S-K$. However, if the price of the underlying asset is lower than the strike price, the call option contract cannot generate a profit. In this case, the value of the function is 0. This call option payoff function plays an important role in calculating the profit or loss of an option contract. It makes it easy to understand the profit or loss situation in an option contract and is also used to price options contracts.

B. Quantum Gates

Quantum gates are fundamental building blocks of quantum circuits that act on single or multiple qubits, allowing for the manipulation of quantum states and enabling reversible quantum computations. Commonly used gates in quantum computing include the H, X, CNOT and Toffoli gates; as illustrated in Figure 1.



(a) H (Hadamard) gate

(b) X (NOT) gate (c) CNOT gate (d) Toffoli gate

Fig. 1: Quantum gates

C. Quantum Adders

Our payoff function utilizes the efficiently implemented quantum adders, namely the Cuccaro [11], Draper [12], Takahashi [13], Gidney adder [10], and Higher Radix architecture adder [14]. Among these, the Cuccaro adder, initially proposed, is an RCA (Ripple Carry Adder) that utilizes one ancilla qubit. To overcome the inherent delay in carry operations within RCAs, the Draper adder, a CLA (Carry Look-ahead Adder), was subsequently proposed. Following that, the Takahashi adder, an ancilla-free RCA, was introduced. The Gidney adder, a relatively recent development, reduces the required number of T gates by half by applying logical AND gates instead of Toffoli gates. Here, halving the cost means that by using logical AND, one of the Toffoli gate pairs used in the adder can be replaced with a logical AND Uncompute operation based on measurements that do not use T gates, so the number of T gates required is halved. The structure of the adder proposed by Gidney in the paper is a mod 2^n which does not compute the most significant bit. However, since our work requires a generic adder, we modified it by adding one output qubit for the top bit $(a+b)_n$ as shown in Figure 2 and used it. From here on, the Gidney adder mentioned in this paper refers to Figure 2. Lastly, the most recent proposal, the Higher Radix architecture adder, is a CLA that integrates techniques from classical carry-lookahead adders and Manchester carry chain adders. In our work, we set it to $radix = 2$. It enables efficient computations by efficiently performing operations. The RCAs have polynomial depth, while the CLAs have logarithmic depth.

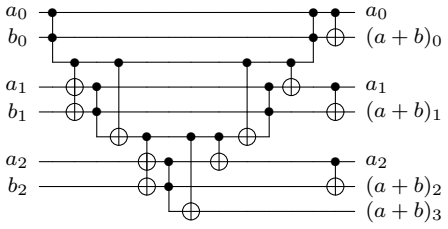


Fig. 2: Modified Gidney adder for $n = 3$

III. Quantum Circuit Implementation of Payoff Function

In this part, we explain the methods we have tested to efficiently implement a quantum circuit for call option payoff function. The function $f_K(S) = \max(S-K,0)$ involves subtracting K from S, outputting the result if it is positive,

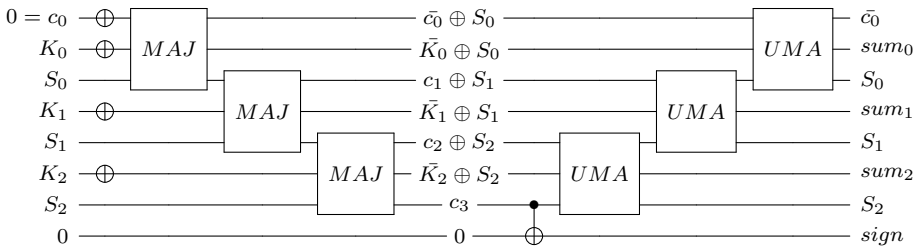


Fig. 3: Implementation of quantum circuit for $S + \bar{K} + 1$ ($n = 3$)

and otherwise outputting 0. We segment this process into two distinct stages. First, calculating the value and sign of $S - K$, and second, deciding whether to return $S - K$ or 0 based on the derived sign. In our quantum circuit implementation, we adopt the concept of two's complement, just like in classical computing. We denote the sizes of S and K as n.

A. Computation of $S - K$

In classical computing, subtraction is achieved by utilizing the circuit for addition. The two's complement and one's complement methods are used for this purpose.

1) Two's Complement (with Ancilla Qubit): For the two binary numbers S and K, the steps for performing subtraction using the two's complement method are as follows. First, compute the two's complement (i.e., $\bar{K} + 1$) of the subtrahend K by inverting each bit and adding 1. Next, add the minuend (S) and the two's complement of the subtrahend just computed. If a carry occurs in this addition, the result is negative (i.e., the most significant bit in the sum of the minuend and the two's complement represents the sign bit). This process can be expressed as $S - K = S + \bar{K} + 1, \because -K = \bar{K} + 1$.

Figure 3 shows the quantum circuit implementation that uses the two's complement subtraction for $n = 3$ bits. The circuit first applies n number of X gates to all qubits of K to take the two's complement. Next, one additional ancilla qubit, labeled c_0 and initialized to 0, is introduced for the '+1' operation. By applying an X gate, we invert its value to 1. This structure is a slight modification of the approach used by Cuccaro [11] for implementing a ripple carry adder, where the initial carry c_0, initialized to 0, is used as the carry bit for the first-bit addition. It calculates carries to c_n through *MAJ* gates and computes the sum using *UMA* gates. We can simply enable the simultaneous addition of three numbers (S, $\bar{K}$, 1) using only one ripple carry adder by setting the initial carry bit to 1.

Overall, one may note that the quantum circuit for the two's complement-based subtraction requires one ancilla qubit, $n + 1$ X gates ($\bar{K}$ and 1), and one quantum ripple carry adder ($S + \bar{K} + 1$).

2) Two's Complement (without Ancilla Qubit): Another expression for $S - K$ can be obtained using the concept of two's complement.

$$S - K = \overline{\bar{S} + K}, \because \bar{K} = -K - 1 \tag{1}$$

To understand why (1) holds, notice the following: $\overline{\bar{S} + K} = \overline{-S - 1 + K} = -(-S - 1 + K) - 1 = S - K$.

Using the expression for $-K$ as shown previously (that uses ancilla qubits), we can derive an expression for $\bar{K}$ as shown in (1) (i.e., $\bar{S} = -S - 1$). Following the same logic, we can also derive that $\overline{\bar{S} + K}$ equals $S - K$. To implement (1) in a quantum circuit, first, apply a X gate to all qubits of S to flip the bits to take the one's complement. Then, the quantum adder is used to perform addition. Finally, the X gate is applied to flip the bits in the result ($\bar{S} + K$) obtained from the previous addition. In other words, the X gate is applied to all qubits except for the carry qubit (i.e., n X gates). To conclude, to implement the circuit as shown in (1), we need a total of $2n$ X gates for $\bar{S}$ and $\overline{\bar{S} + K}$, along with one quantum adder ($\bar{S} + K$).

We have explored two distinct implementations for $S - K$, and ultimately, the approach of (1) demonstrates greater efficiency by eliminating the need for an ancilla qubit in handling the '+1' addition. Therefore, we apply this method for implementing payoff function.

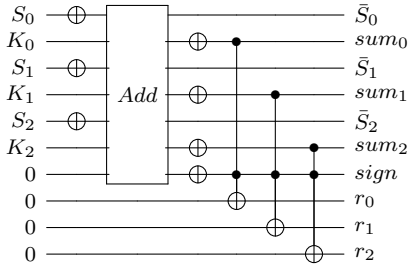


Fig. 4: Space-efficient implementation for payoff ($n = 3$)

B. Implementation of $f_K(S) = max(S - K, 0)$

In contrast to classical computers, quantum computer circuits must be reversible, ensuring a one-to-one mapping between inputs and outputs. However, the function $max(S - K, 0)$ does not exhibit a one-to-one correspondence; it is one-to-one for positive values but consistently evaluates to 0 for negative values. To make payoff function have a one-to-one correspondence, we use additional registers (i.e., qubits) to store the result. We add a register to store the result value and the remaining space stores values for reversibility.

The method is as follows: First, we add n qubits initialized to 0, where the final result (r_i) will be stored. The next step involves using a sign qubit ($sign$) to determine the value of r_i. In the case of a negative ($sign = 1$), the output r_i should be set to 0, while for a positive, the output should be equal to the subtraction result sum_i. To do this, the $max(S-K, 0)$ circuit

TABLE I: Comparison of adder resources

$^*w(n) = n - \sum_{k=1}^{\infty} \lfloor \frac{n}{2^k} \rfloor$

Adder		Qubits	# Toffoli	Toffoli-depth
Cuccaro [11]	UMA 2CNOT	$2n+2$	$2n$	$2n$
	UMA 3CNOT		$2n-1$	$2n-1$
Draper [12]	in-place	$4n - w(n) - \lfloor \log n \rfloor$	$10n - 3w(n) - 3w(n-1)$ $-3\lfloor \log n \rfloor - 3\lfloor \log (n-1) \rfloor - 7$	$\lfloor \log n \rfloor + \lfloor \log (n-1) \rfloor$ $+\lfloor \log \frac{n}{3} \rfloor + \lfloor \log \frac{n-1}{3} \rfloor + 8$
	out-of-place	$4n - w(n) - \lfloor \log n \rfloor + 1$	$5n - 3w(n) - 3\lfloor \log n \rfloor - 1$	$\lfloor \log n \rfloor + \lfloor \log \frac{n}{3} \rfloor + 4$
Takahashi [13]		$2n+1$	$2n-1$	$2n-1$

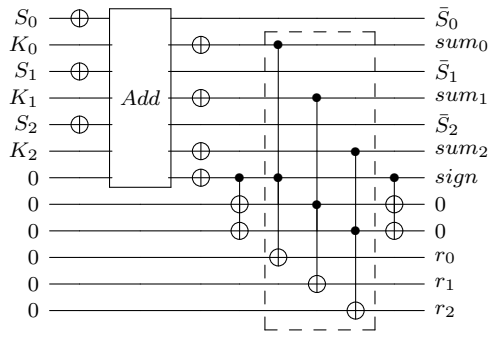


Fig. 5: Time-efficient implementation for payoff ($n = 3$)

starts by inverting the value of $sign$. This can be implemented by applying a X gate to the $sign$. Then $sign$ and sum_i are ANDed and XORed with r_i. This can be implemented with Toffoli gates.

We propose two versions of payoff for optimization: *Space-efficient* and *Time-efficient* versions. Figure 4 and Figure 5 illustrate the implementation of the Qubit optimization circuit and Toffoli-depth optimization circuit when $n = 3$ bits, respectively. In order to show the circuit of payoff in the simplest way, it is assumed that the Takahashi adder [13], which does not use ancilla, is used for Add operation.

The Toffoli-depth optimization circuit utilizes ancilla qubits in addition to the qubit optimization circuit to parallelize Toffoli gate operations. We add $n-1$ qubits and copy the state of $sign$. As a result, Toffoli-depth can be reduced to 1. Such types of implementations result in T-depth optimization when applying Toffoli decomposition.

IV. Resource Comparison

A. Resource Comparison of Various Quantum Adders

The Gidney adder is a modified Cuccaro-series adder that replaces pairs of Toffoli gates with logical AND gates. Similarly, the higher radix adder also replaces logical AND gates in some of its Toffoli gates. Therefore, these two adders will be discussed in Section IV-C).

Cuccaro adder has two versions of the UMA operation: 2 CNOT and 3 CNOT gates, with the latter being suitable for parallel implementation. Draper adder also has two versions, depending on where the operation results are stored, in-place and out-of-place. Table I presents a resource comparison of the Cuccaro, Draper, and Takahashi adders. The resources are compared based on the number of qubits, Toffoli gates, and Toffoli-depth for each adder. The asymptotic notation in Table I applies when $n \geq 4$. Takahashi adder uses the fewest qubits, and Draper out-of-place has the smallest Toffoli-depth. When $n \geq 9$, Draper in-place has a smaller Toffoli-depth than Cuccaro UMA 3 CNOT gates.

TABLE II: Resource comparison of two versions of payoff function

	Qubits	# X	# CNOT	# Toffoli	Toffoli-depth
$f1$	n	$2n+1$	0	n	n
$f2$	$2n-1$		$2n-2$		1

B. Resource Comparison of Payoff Function

Table II presents a resource comparison of two versions of payoff function. *Space-efficient* and *Time-efficient* are referred to as $f1$ and $f2$, respectively. The resources are compared based on the number of Toffoli gates, and Toffoli-depth, excluding the resources required for Add operation. $f1$ has the Toffoli-depth of n but does not use ancilla qubits, while $f2$ has the constant Toffoli-depth of 1 but uses ancilla qubits and CNOT gates. Both versions have a trade-off relationship, so the choice depends on the computing environment used to implement this Quantum Arithmetic. Based on the resources specified in these two tables, Section IV-C applies Toffoli decomposition to payoff function and adders to perform resource comparisons.

C. Resource Comparison of Various Toffoli Decompositions

The Toffoli gate can be decomposed into a combination of Clifford gates and T gates. This decomposition process represents the Toffoli gate using a set of more fundamental Clifford gates. Such decomposition is important for quantum algorithm design, optimization of gate combinations, error correction, and other purposes in quantum computing. Table III represents the versions of Toffoli Decomposition (TD) used in our work, where each version is named after the number of Ancilla qubits employed. As the number of Ancilla qubits decreases, the T-depth increases, indicating a trade-off relationship. Logical AND operation is composed of Compute and Uncompute operations, and a pair of Toffoli gates where the Toffoli gate used later uncompute the Toffoli gate used earlier can be replaced with the Comp and Uncomp of logical AND. Since the Uncomp operation does not utilize the T gate, the

TABLE III: Resource of four versions of Toffoli Decomposition (TD)

TD		Ancilla	# T	T-depth
Zero [15]		0	7	3
One [5]		1		2
Four [15]		4		1
logical AND	COMP	1	4	2
	UNCOMP	0	0	0

resources related to the T gate can be reduced by half. For non-paired Toffoli gates, a single Toffoli gate can be replaced by using one ancilla qubit and the Comp and Uncomp operations of logical AND gate.

In Table III, the T-depth of the Compute operation of logical AND gate is indicated as 2; however, for n greater than 2, the generation of T states can be processed in parallel. Therefore, the Gidney adder has a T-depth of $n+1$, and the T-depth of the adder with Toffoli gates replaced by logical AND gates is calculated by adding 1 to the used Compute operations' depth. Cuccaro and Takahashi adders become Gidney adders when the logical AND operation is applied instead of Toffoli gate. Thus, it is not necessary to apply logical AND operations to the Cuccaro and Takahashi adders in Table IV. Instead, using the Gidney adder would be sufficient. Additionally, by default or due to TD, the adder that utilizes ancillas resets them to 0 after Add operation, allowing for reuse in the max operation. However, the adder with logical AND replacements does not reuse ancilla since measurements are performed during the uncompute phase. Table IV shows the detailed analysis results, and Figure 6 is based on this and visualizes the trade-off between qubit count and T-depth, focusing on the resources used for zero and logical AND Toffoli decompositions. The asymptotic notation represented in the table can be applied from $n \geq 4$, as before, and when the notation becomes too long, it is shown by replacing it with symbols like $F(n)$. As evident from the graph, in terms of qubit count, the CLA family shows a higher count. This is because CLA processes Toffoli gates in parallel, resulting in the usage of ancilla by TD being equal to the maximum number of Toffoli gates that can be processed simultaneously. Particularly, the Draper and Higher Radix adders, which apply logical AND gates, show the highest counts. This is because ancilla cannot be reused, leading to the usage of ancilla equal to half the number of Toffoli gates that do pairs, plus the number of Toffoli gates that do not pair. However, when focusing on the T-count, it is noticeable that Draper's in-place and out-of-place implementations with logical AND have a count of less than half. Furthermore, when considering the T-depth metric, the advantages become more prominent. The CLA adders with logical AND show the lowest T-depth, and the gap becomes more significant in $F2$ where the T-depth remains constant at 1. The Gidney adder has the smallest T-count, and considering all four graphs, it can be observed that, on average, it utilizes the least resources. Consequently, to maximize the advantage of having low T-depth in the payoff function, it is recommended to use the Draper out-of-place adder with the lowest T-depth and apply logical AND to the circuit.

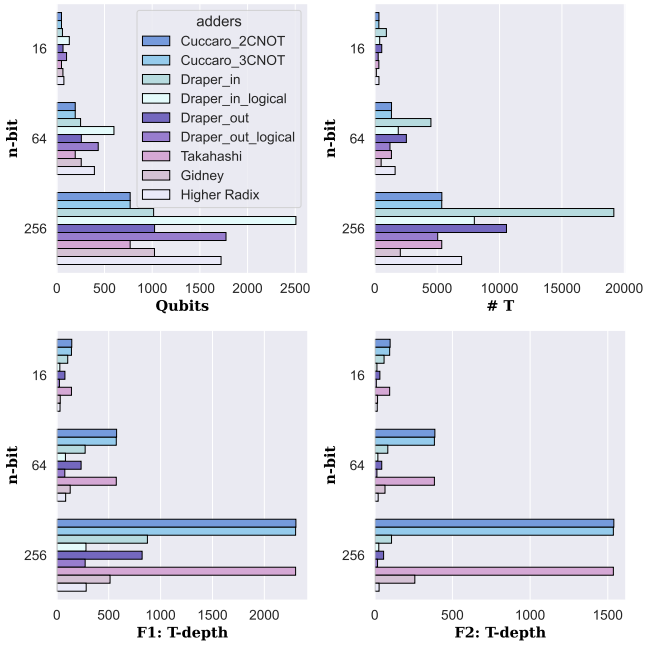


Fig. 6: Comparison in terms of qubit count and T-depth trade-off of payoff function

V. Conclusion

In this study, we have successfully proposed and implemented a quantum circuit for the call option payoff function, a key component in quantum finance. This optimized circuit, designed with careful consideration of resource efficiency, provides a potential solution for faster and more effective computation of this financial function on quantum hardware.

Our contributions include an optimized quantum circuit design for payoff function, a space-efficient implementation reducing the qubit count, a time-efficient implementation minimizing the circuit depth, and a detailed comparison of various adders and Toffoli decompositions.

Our quantum circuit design utilizes two's complement subtraction for the operation of $S - K$ and a supplementary register to ensure reversibility, overcoming the fundamental constraint of quantum circuits. Our proposed implementations effectively optimize the depth and number of qubits, which enables efficient execution on quantum hardware. Moreover, we comprehensively compared different quantum adders and Toffoli decompositions, presenting a varied selection for optimal execution depending on the specific quantum computing environment. Future research should focus on further optimizing the quantum circuit design and making them more robust against noise and errors.

VI. Acknowledgement

This research is supported by the National Research Foundation, Singapore under its Quantum Engineering Programme Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore. Also, This work was supported by Institute for

TABLE IV: Comparison of Payoff Function resources based on various TD

$*w(n) = n - \sum_{k=1}^{\infty} \lfloor \frac{n}{2^k} \rfloor \quad *F(n) = \lfloor \log n \rfloor + \lfloor \log \frac{n}{3} \rfloor$

Adder		Payoff	TD	Qubits	# T	T-depth
Cuccaro	UMA 2CNOT	f1	Zero	$3n+1$	$21n$	$9n$
			One	$3n+2$		$6n$
			Four	$3n+5$		$3n$
		f2	Zero	$4n$		$6n+3$
			One	$4n+1$		$4n+2$
			Four	$4n+4$		$2n+1$
	UMA 3CNOT	f1	Zero	$3n+1$	$21n-7$	$9n-3$
			One	$3n+2$		$6n-2$
			Four	$3n+5$		$3n-1$
		f2	Zero	$4n$		$6n$
			One	$4n+1$		$4n$
			Four	$4n+4$		$2n$
Draper	In-place	f1	Zero	$4n - w(n) - \lfloor \log n \rfloor$	$77n - 21w(n) - 21w(n-1)$ $-21\lfloor \log n \rfloor - 21\lfloor \log (n-1) \rfloor - 49$	$3n + 3F(n) + 3F(n-1) + 24$
			One	$5n - w(n) - \lfloor \log n \rfloor$		$2n + 2F(n) + 2F(n-1) + 16$
			Four	$8n - w(n) - \lfloor \log n \rfloor$		$n + F(n) + F(n-1) + 8$
			logical AND	$10n - 2w(n) - 2w(n-1)$ $-2\lfloor \log n \rfloor - 2\lfloor \log (n-1) \rfloor - 5$	$32n - 8w(n) - 8w(n-1)$ $-8\lfloor \log n \rfloor - 8\lfloor \log (n-1) \rfloor - 20$	$n + F(n) + \lfloor \log (n-1) \rfloor + 4$
		f2	Zero	$4n$	$77n - 21w(n) - 21w(n-1)$ $-21\lfloor \log n \rfloor - 21\lfloor \log (n-1) \rfloor - 49$	$3F(n) + 3F(n-1) + 27$
			One	$5n$		$2F(n) + 2F(n-1) + 18$
			Four	$8n$		$F(n) + F(n-1) + 9$
			logical AND	$11n - 2w(n) - 2w(n-1)$ $-2\lfloor \log n \rfloor - 2\lfloor \log (n-1) \rfloor - 6$	$32n - 8w(n) - 8w(n-1)$ $-8\lfloor \log n \rfloor - 8\lfloor \log (n-1) \rfloor - 20$	$F(n) + \lfloor \log (n-1) \rfloor + 5$
	Out-of-place	f1	Zero	$4n+1$	$42n - 21w(n) - 21\lfloor \log n \rfloor - 7$	$3n + 3F(n) + 12$
			One	$5n+1$		$2n + 2F(n) + 8$
			Four	$8n+1$		$n + F(n) + 4$
			logical AND	$7n - 2w(n) - 2\lfloor \log n \rfloor$	$20n - 8w(n) - 8\lfloor \log n \rfloor - 4$	$n + F(n) + 3$
		f2	Zero	$5n$	$42n - 21w(n) - 21\lfloor \log n \rfloor - 7$	$3F(n) + 15$
			One	$6n$		$2F(n) + 10$
			Four	$9n$		$F(n) + 5$
			logical AND	$8n - 2w(n) - 2\lfloor \log n \rfloor - 1$	$20n - 8w(n) - 8\lfloor \log n \rfloor - 4$	$F(n) + 4$
Takahashi		f1	Zero	$3n+1$	$21n-7$	$9n-3$
			One	$3n+2$		$6n-2$
			Four	$3n+5$		$3n-1$
		f2	Zero	$4n$		$6n$
			One	$4n+1$		$4n$
			Four	$4n+4$		$2n$
Gidney		f1	logical AND	$4n$	$8n$	$2n+1$
		f2		$5n-1$		$n+2$
Higher Radix architecture adder		f1	logical AND	$5n - 17 + 4\lceil \frac{n}{2} \rceil - (n-1)\,(mod\,2)$ $-4w(\lceil \frac{n}{2} \rceil - 1) - 4\lfloor \log(\lceil \frac{n}{2} \rceil - 1) \rfloor$	$32\lceil \frac{n}{2} \rceil + 12n - 52 - 4(n-1)\,(mod\,2)$ $-12w(\lceil \frac{n}{2} \rceil - 1) - 12\lfloor \log(\lceil \frac{n}{2} \rceil - 1) \rfloor$	$2\lfloor \log(\lceil \frac{n}{2} \rceil - 1) \rfloor + \lfloor \log(\frac{1}{3}(\lceil \frac{n}{2} \rceil - 1)) \rfloor + n + 10$
		f2		$6n - 18 + 4\lceil \frac{n}{2} \rceil - (n-1)\,(mod\,2)$ $-4w(\lceil \frac{n}{2} \rceil - 1) - 4\lfloor \log(\lceil \frac{n}{2} \rceil - 1) \rfloor$		$2\lfloor \log(\lceil \frac{n}{2} \rceil - 1) \rfloor + \lfloor \log(\frac{1}{3}(\lceil \frac{n}{2} \rceil - 1)) \rfloor + 11$

Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (<Q|Crypton>, No.2019-0-00033, Study on Quantum Security Evaluation of Cryptography based on Computational Quantum Complexity)

REFERENCES

[1] P. Rebentrost, B. Gupt, and T. R. Bromley, "Quantum computational finance: Monte carlo pricing of financial derivatives," *Physical Review A*, vol. 98, no. 2, p. 022321, 2018.

[2] R. Orús, S. Mugel, and E. Lizaso, "Quantum computing for finance: Overview and prospects," *Reviews in Physics*, vol. 4, p. 100028, 2019.

[3] D. Herman, C. Googin, X. Liu, A. Galda, I. Safro, Y. Sun, M. Pistoia, and Y. Alexeev, "A survey of quantum computing for finance," *arXiv preprint arXiv:2201.02773*, 2022.

[4] A. Saha, T. Chatterjee, A. Chattopadhyay, and A. Chakrabarti, "Intermediate qutrit-based improved quantum arithmetic operations with application on financial derivative pricing," *arXiv preprint arXiv:2205.15822*, 2022.

[5] M. Amy, D. Maslov, and M. Mosca, "Polynomial-time t-depth optimization of clifford+ t circuits via matroid partitioning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1476–1489, 2014.

[6] X. Zhou, D. W. Leung, and I. L. Chuang, "Methodology for quantum logic gate construction," *Physical Review A*, vol. 62, no. 5, p. 052316, 2000.

[7] D. Gosset, V. Kliuchnikov, M. Mosca, and V. Russo, "An algorithm for the t-count," *arXiv preprint arXiv:1308.4134*, 2013.

[8] S. J. Devitt, A. M. Stephens, W. J. Munro, and K. Nemoto, "Requirements for fault-tolerant factoring on an atom-optics quantum computer," *Nature communications*, vol. 4, no. 1, p. 2524, 2013.

[9] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, no. 3, p. 032324, 2012.

[10] C. Gidney, "Halving the cost of quantum addition," *Quantum*, vol. 2, p. 74, 2018.

[11] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A new quantum ripple-carry addition circuit," *arXiv preprint quant-ph/0410184*, 2004.

[12] T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore, "A logarithmic-depth quantum carry-lookahead adder," *arXiv preprint quant-ph/0406142*, 2004.

[13] Y. Takahashi, S. Tani, and N. Kunihiro, "Quantum addition circuits and unbounded fan-out," *arXiv preprint arXiv:0910.2530*, 2009.

[14] S. Wang, A. Baksi, and A. Chattopadhyay, "A higher radix architecture for quantum carry-lookahead adder," *arXiv preprint arXiv:2304.02921*, 2023.

[15] P. Selinger, "Quantum circuits of t-depth one," *Physical Review A*, vol. 87, no. 4, p. 042302, 2013.