The Quantum Cyclic Rotation Gate

Arianna Pavone[1,†], Caterina Viola[2,*,†]

[1]Università degli Studi di Palermo, Dipartimento di Matematica e Informatica
[2]Università degli Studi di Catania, Dipartimento di Matematica e Informatica

Abstract

A *circular shift operator* (or *cyclic rotation gate*) ROT_k applies a rightward (or leftward) shift of k positions to a register of n qubits so that the element at position x is moved to position $(x+k) \mod n$. While it is known that there exists a quantum rotation operator that can be implemented in $\mathcal{O}(\log(n))$-time, through the repeated parallel application of the elementary Swap operators, there is no systematic procedure that concretely constructs the quantum operator ROT for variable size n of the quantum register and a variable parameter k. We show a concrete implementation of the cyclic rotation operator (denoted ROT) in a quantum circuit model of computation. The depth of the obtained circuit implementing the cyclic rotation operator is upper-bounded by $\log n$; therefore, the operator ROT_k can be implemented in $\mathcal{O}(\log(n))$-time. When the parameter k dictating the magnitude of the rotation is a power of 2, namely when $k = 2^m$ for some 2, the depth of the circuit is exactly $\log(n) - \log(k)$.

Keywords

Quantum Circuits, Text Processing, Quantum Rotation

1. Introduction

Quantum computing represents an avant-garde domain within the realm of computer science, where the intricate principles of quantum mechanics are harnessed to engineer formidable computing systems that manifest striking deviations from classical counterparts. In stark contrast to classical computers, which process information using discrete binary bits constrained to exclusively one of the states 0 and 1, quantum computing harnesses the power of quantum bits, or qubits, which effortlessly inhabit superpositions of multiple states. Moreover, the entangled configuration of two or more qubits bestows upon them the extraordinary ability to execute synchronized operations, transcending the computational efficiency of classical bits. These distinctive attributes endow quantum computers with a momentous advantage over their classical counterparts.

There are several models of quantum computation, with their own advantages and challenges. These include the *adiabatic model* [1], based on the adiabatic theorem of quantum mechanics, the *topological model* [2], based on the principles of topological quantum field theory, and the *measurement-based model* [3], where computation is performed by making measurements on

Italian Conference on Theoretical Computer Science (ICTCS), September 13–15, 2023, Palermo, Italy
*Corresponding author. This author is funded by ICSC – Centro Nazionale di Ricerca in High-Performance Computing, Big Data and Quantum Computing, co-founded by European Union - NextGenerationEU.
†These authors contributed equally.
ariannamaria.pavone@unipa.it (A. Pavone); caterina.viola@unict.it (C. Viola)
0000-0002-8840-6157 (A. Pavone); 0000-0002-7312-5002 (C. Viola)

an entangled resource state known as a cluster state. However, the *circuit model* [4] is the most common and widely used model in quantum computing. It represents computations as sequences of quantum gates that act on qubits to manipulate and transform quantum information. Operations are executed sequentially, and measurements are performed to extract classical information from the quantum system.

The complexity of a quantum circuit can be measured in several ways. One of these is the number of gates used within the circuit to manipulate the qubits, but since it is often possible to run two or more gates in parallel when operating on disjoint registers, a more appropriate measure is the depth of the circuit, i.e., the number of steps performed before the output of the circuit is obtained.

The design and simulation of efficient quantum circuits capable of solving specific tasks, including through the use of artificial intelligence models [5], is a particularly active area of research in recent years.

In this paper, we address the problem constructing of a *quantum cyclic shift* (or *cyclic rotation*) operator, in the quantum circuit model of computation.

Given a vector x of length n and an input parameter $s < n$, a cyclic rotation of a vector is a transformation that shifts the elements of the vector in circular positions, while maintaining their relative order. In other words, each element of the vector is moved by s positions, and the last s elements are brought back to the first positions of the vector.

Cyclic rotations of vectors have various applications, including, for instance, image and signal processing where they can be employed to perform cyclic shifts on images or signals, such as image rolling [6] or time delay of a signal. Cyclic rotations can be also used to design efficient algorithms sorting data [7]. In addition, sequences admitting cyclic rotations are also revalent in various biological contexts, including viruses [8, 9] and bacteria [10]. Thus, the analysis of organisms with a cyclic structure can benefit from algorithms designed for strings that allow for cyclic rotations [11].

In the field of quantum computation, cyclic rotation of a register has been effectively used in solutions for text processing, and specifically in string matching. The recent algorithm by Niroula and Nam [12] cleverly uses cyclic rotations of the registers containing the input strings to achieve a superimposition of all their possible alignments and to perform a parallel comparison against the pattern. This idea was later used by Cantone *et al.* [13] to efficiently solve the string matching problem allowing for swaps of adjacent characters.

Since a cyclic rotation of a vector of n elements of s positions to the right consists essentially of a permutation of the input vector in which each element of position i is moved to position $(i+s) \mod (n)$, it is easy to construct a classical procedure capable of achieving such a rotation in linear time.

In quantum computation, on the other hand, it is possible to exploit the parallelism of gate execution within a circuit to achieve significant speed-up. Niroula and Nam [12] provide insight into the fact that such a circuit can be executed in time $\mathcal{O}(\log(n))$. The basic idea is that at each step of the algorithm that accomplishes the permutation, it is possible to place at least half of the qubits that still need to be moved to their final position. Since the number of qubits to be placed decreases by at least half at each iteration, $\mathcal{O}(\log(n))$ steps are needed to achieve the target permutation. However, they do not provide any procedure explaining how to construct this quantum operator, nor do they provide a more formal proof of its complexity.

In an attempt to fill the gap, this paper aims to provide a precise method for the construction of a circular rotation operator for a vector of dimension n as the parameter s, indicating the shift relative to the rotation, varies. As far as we know, this is the first work offering such a procedure. A proof of the correctness of our procedure is also provided, along with an analysis of the time complexity of the resulting circuit. We believe that this result may be of interest to the scientific community concerned with the design and simulation of quantum algorithms, especially in the area of text processing.

The paper is organized as follows. In Section 2 we recall some basic notions, introduce some useful notations adopted along the paper and give a more formal definition of the problem. In Section 3 we present a solution for the specific case where $s = 2^p$ for some $p \in \mathbb{N}$, prove its correctness and discuss its complexity analysis. In Section 4 we extend our solution to the general case. Finally we draw our conclusions in Section 5.

2. Preliminaries and Definition of the Problem

The fundamental unit in quantum computation is the *qubit*. A qubit is a coherent superposition of the two orthonormal basis states, which are denoted by $|0\rangle$ and $|1\rangle$, using the conventional *bra–ket* notation. The mathematical formulation for a qubit $|\psi\rangle$ is then a linear combination of the two basis states, i.e., $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where the values α and β, called *amplitudes*, are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$, representing the probability of finding the qubit in the state $|0\rangle$ or $|1\rangle$, respectively, when measured. A quantum measurement is the only operation through which information is gained about the state of a qubit, however causing the qubit to collapse to one of the two basis states. The measurement of the state of a qubit is irreversible, meaning that it irreversibly alters the magnitudes of α and β. If b is a binary value, equal to 0 or 1, we use the symbol $|b\rangle$ to indicate the qubit in the corresponding basis state, $|0\rangle$ or $|1\rangle$, respectively. Multiple qubits taken together are referred to as *quantum registers*. A quantum register $|q\rangle = |q_0, q_1, .., q_{n-1}\rangle$ of n qubits is the tensor product of the constituent qubits, i.e., $|q\rangle = \bigotimes_{i=0}^{n-1} |q_i\rangle$.

If k is an integer value that can be represented by a binary string of length n, we use the symbol $|k\rangle$ to denote the register of n qubits such that $|k\rangle = \bigotimes_{i=0}^{n-1} |k_i\rangle$, where $|k_i\rangle$ takes the value of the i-th least significant binary digit of k. For example, the quantum register $|9\rangle$ with 4 qubits is given by $|9\rangle = |1\rangle \otimes |0\rangle \otimes |0\rangle \otimes |1\rangle$. The mathematical formulation of a quantum register is then $|q\rangle = \sum_{k=0}^{2^n-1} \alpha_k|k\rangle$, where the values α_k represent the probability of finding the register in the state $|k\rangle$ when measured, with $\sum_{k=0}^{2^n-1} |\alpha_k|^2 = 1$.

The model of computation we adopt in this paper is that of *reversible circuits*. Circuits are networks composed of wires that carry qubit values to *gates* that perform elementary operations on qubits. The qubits move through the circuit in a linear fashion, where the input values are written onto the wires entering the circuit from the left side, while the output values are read-off the wires leaving the circuit on the right side. At every time step, each wire can enter at most one gate. In the definition of a circuit, it is often necessary to include *ancillæ* qubits, which are needed to achieve some specific tasks in computation that otherwise could not be achieved.

For the circuit model of computation, a natural measure of complexity is the *number of gates* used in the circuit. If we assume the circuit as being divided into a sequence of discrete

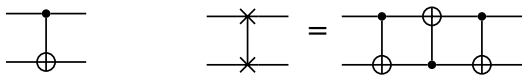

Figure 1: The representation of the CNOT and Swap gates. The Swap gate corresponds to three CNOT gates.

time-steps, where the application of a single gate requires a single time-step, another measure of complexity is the *depth* of the circuit, which is the total number of required time-steps. We observe that this is not necessarily the same as the total number of gates in the circuit, since gates that act on disjoint qubits can often be applied in parallel.

There is a variety of quantum operators capable of operating on quantum registers to perform widely-ranging manipulations. Here we simply list the two gates that will be used in this work: the CNOT gate and the Swap gate.

The *controlled* NOT *gate* (or CNOT) is a quantum logic gate operating on a register of two qubits $|q_0, q_1\rangle$. If the control qubit $|q_0\rangle$ is set to 1, it inverts the target qubit $|q_1\rangle$, otherwise all qubits stay the same. Formally, it maps $|q_0, q_1\rangle$ to $|q_0, q_0 \oplus q_1\rangle$. The *Swap gate* is a two-qubit operator: expressed in basis states, it swaps the state of the two qubits $|q_0, q_1\rangle$ involved in the operation, mapping them to $|q_1, q_0\rangle$. Interestingly, the swap gate can be achieved by the application of three CNOT operators.

Fig. 1 shows the representation of the CNOT and Swap gates.

A *circular shift operator* (or rotation operator) ROT_k applies a rightward shift of k positions to a register of n qubits so that the element at position x is moved to position $(x + k) \mod n$. In other words, the elements whose position exceeds the size n of the register are moved, in a circular fashion, to the first positions of the register. Formally, the operator ROT_k applies the following permutation

$$|q_0, q_1, \ldots, q_{n-1}\rangle \longrightarrow |q_{n-s}, q_{n-s+1}, \ldots, q_{n-1}, q_0, q_2, \ldots, q_{n-s-1}\rangle.$$

Throughout the document we assume that the size n of the qubit to be rotated is of the form 2^p for some $p \in \mathbb{N}$, observe that this assumption is not restrictive since for quantum registers made of qubits this is always the case; nevertheless, in Section 4.1 we show how to proceed seamlessly in the case that the size of the target register is not a power of 2.

3. An Algorithm for $k = 2^m$

In this section we describe an algorithm for cyclically rotating a quantum register $|q\rangle$ of n qubits of k positions, with $0 < k < n$. We recall that we assume $n = 2^p$ for some $p \in \mathbb{N}$. Without loss of generality, we assume that the rotation operator shifts the qubits to the right. Leftward rotations can occur symmetrically with respect to what is described in this paper.

We first consider the case in which the parameter controlling the rotation is of the form $k = 2^m$, with $m < p$. Actually, it is enough to assume $k = \frac{n}{2^h}$ for some $h\colon 1 \leq h \leq p - 1$. The pseudocode of the quantum procedure performing the cyclic rotation is presented in Algorithm 1.

ALGORITHM 1: Algorithm for $k = 2^m$

Input: a n-qubit register $q := \bigotimes_{x=0}^{n-1} |q_x\rangle$, and $k := 2^m$.
Output: a n-qubit register $q' := \bigotimes_{x=0}^{n-1} |q_{x+k \mod (n)}\rangle$.

```
1 for i = 1, ..., log(n) − log(k); i++ do
2     for j = 0, ..., n/(2^i k) − 1; j++ do
3         for x = jk2^i, ..., (j2^i + 1)k − 1; x++ do
4             Swap(q_x, q_{x+2^{i−1}k})
```

The procedure performs a permutation of the qubits contained in the input quantum register. This is done by means of a sequence of swap operations that exchange the positions of two qubits inside the register. During the execution of the algorithm we distinguish qubits having reached their final position, which we indicate with the term *placed qubits*. Conversely, qubits having not yet been placed correctly are indicated by the term *out-of-place qubits*.

In brief, the algorithm works as follows. At the beginning of the procedure the register $|q\rangle$ contains n not-in-place qubits and no placed qubit. Then, an iterative cycle starts, where at each iteration step, the algorithm selects half of the remaining not-in-place qubits and swaps them to their final positions. This means that the procedure stops in at most $\log(n)$ steps.

The worst case is obtained when each swap operation places only one of the two involved qubits in its final place, terminating in $\log(n)$ steps. When each swap succeeds in placing both qubits in their final positions, then the algorithm obtains its best case ($k = \frac{n}{2}$), terminating in constant time.

We now go into more detail on the design of Algorithm 1. It is based on a main iterative loop (line 1) that runs $\log(n) - \log(k)$ times[1].

In the first iteration (namely for $i = 1$), the algorithm decomposes the register $|q\rangle$ into $\frac{n}{2k}$ intervals, each of size $2k$ (line 2). Let I_j, for $0 < j < \frac{n}{2k}$, be the j-th interval into which the register $|q\rangle$ has been divided. The algorithm divides each interval I_j into two halves of size k. In this context, let I_j^ℓ be the left half of the interval I_j and let I_j^r be the right half of the same interval. The algorithm operates by swapping the qubits in I_j^ℓ with the corresponding qubits in I_j^r (line 3). Specifically it applies a Swap to the pair of qubits (q_x, q_{x+k}) for every x corresponding to a position in I_j^ℓ, that is $x \in \{2jk, \ldots, (2j+1)k - 1\}$. It is important to stress the fact that the algorithm performs these swaps in parallel for each $j \in \{0, \ldots, \frac{n}{2k} - 1\}$.

Since this operation shifts the qubits in I_j^ℓ exactly k positions to the right, after the first iteration, half of the qubits are correctly placed. It is immediate to see, indeed, that the algorithm correcly placed the qubits that have been moved to positions $x + k$, for $x \in \{2jk, \ldots, (2j + 1)k - 1\}$ and for $j \in \{0, \ldots, \frac{n}{2k} - 1\}$.

We can prove (see Section 3.1) that if $k = \frac{n}{2}$, that is if $i = 1$, we do not need further iterations as also the qubits at positions q_x, for $x \in \{2jk, \ldots, (2j+1)k - 1\}$ and for $j \in \{0, \ldots, \frac{n}{2k} - 1\}$, are correctly placed and the algorithm correctly terminates. Otherwise, the algorithm starts a new iteration.

The interval decomposition that we described in the first iteration can be generalized in subsequent iterations of the main for loop as follows. In the ith step, being $1 < i \leq \log(n) -$

[1]We'll discuss why the iterative loop on line 1 runs $\log(n) - \log(k)$ times later.

$\log(k)$), the algorithm decomposes the register $|q\rangle$ into $\frac{n}{2^i k}$ intervals, each of size $2^i k$ (line 2).

Now observe that the first k elements of the left interval I_j^ℓ are in that position as a result of a swap of the previous iteration. We are therefore dealing with not-in-place qubits. The same is true for the first k qubits of the interval I_j^r. Thus, the algorithm operates by swapping the qubits in the first k positions of I_j^ℓ with the corresponding qubit in the first k positions of I_j^r (line 3). This is done to forbid swaps of qubits that are already correctly placed.

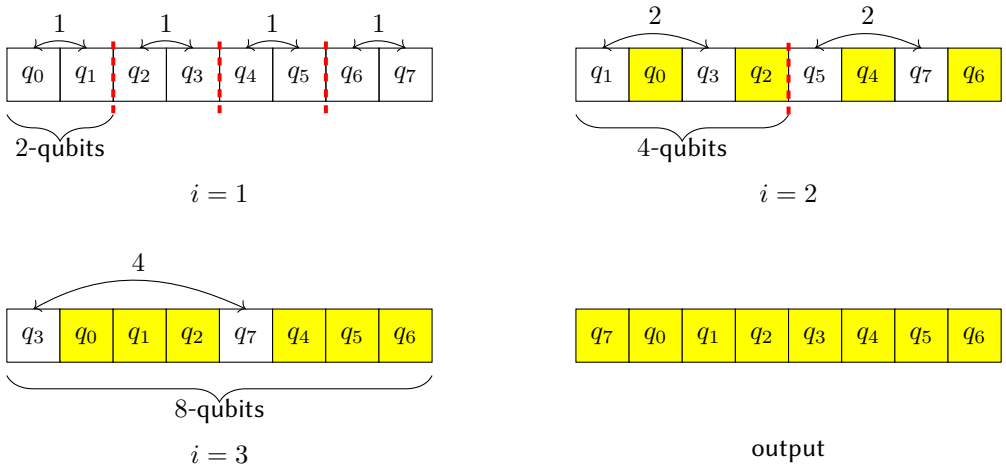

Figure 2: Illustration of the iterations of the algorithm implementing the circular shift operator for a register of 8 qubits in which a rotation of 1 position is performed. The coloured qubits are *placed qubits*.

More formally, the algorithm applies a swap to the pair of qubits $(q_x, q_{x+2^{i-1}k})$ for every x corresponding to a position in the first half of I_j, that is $x \in \{2^i jk, \ldots, (2^i j+1)k-1\}$. Also in this case the algorithm performs these swaps in parallel for each $j \in \{0, \ldots, \frac{n}{2^i k} - 1\}$. Figure 2 illustrates the iteration of the algorithm implementing the circular shift operator for a register of 8 qubits in which a rotation of 1 position is performed. Figure 3 provides an illustration of the application of the circular shift operator as prescribed by Algorithm 1, for a register of 8 qubits in which a rightward circular shift of magnitude 1, 2, and 4 is performed. In the representation of each operator, the time-steps, within which the swaps are executed in parallel, have been framed. Each time-step is associated with a label $\text{ROT}_k^{[i]}$, where k represents the shift amount and i represents the number of the iterative step.

Regarding the computational complexity, we observe that in a quantum circuit model of computation the **for**-loops at lines 2 and 3 of Algorithm 1 can be executed in parallel. To see this it is enough to check that the aim of line 2 is to partition the register in disjoint intervals, while the aim of lines 3-4 is to swap disjoint qubits in such intervals. More precisely, each iteration of line 3 refers to a specific one of the intervals individuated in line 2; because, the intervals are disjoint, the whole computation described at lines 2-4 happens in parallel in one time-step.

It follows that the running time of Algorithm 1 is $\log(n) - \log(k)$, which is $\mathcal{O}(\log(n))$.

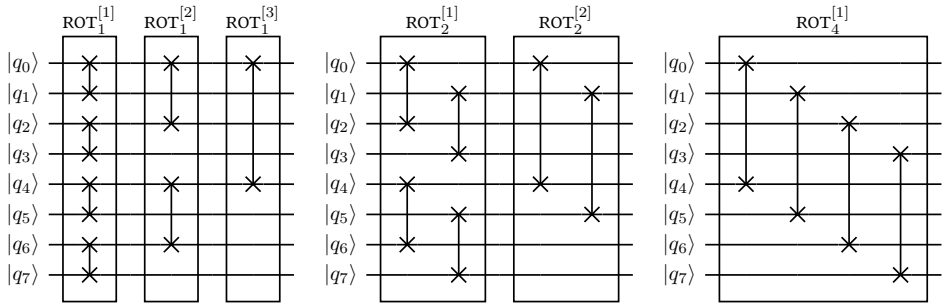

Figure 3: The application of the circular shift operator for a register of 8 qubits in which a rotation of 1, 2, and 4 positions is performed, respectively.

3.1. Correctness of Algorithm 1

In this section, we prove the correctness of Algorithm 1. We start by making the following considerations.

1. At the ith iteration, with $1 < i \leq \log(n) - \log(k)$, a qubit is not involved in any Swap if and only if it has already been correctly shifted in a previous iteration, i.e. it is a placed qubit.
2. After $\log(n) - \log(k)$ iterations of Algorithm 1, every qubit is correctly cyclically shifted by k positions rightwardly, i.e. the jth qubit has been placed at position $j + k \mod (n)$.

For $i \in \{1, \ldots, \log(n) - \log(k)\}$, the symbol q_x^i denotes the xth qubit of the register $|q\rangle$ at the end of the ith iteration of the algorithm; also, we set $q_x^0 := q_x$, and $|q^i\rangle = \bigotimes_{x=0}^{n} q_x^i$, accordingly.

The correctness of Algorithm 1 immediately follow from the next lemma.

Lemma 3.1. *If $1 \leq i \leq \log(n) - \log(k)$, at the end of the ith iteration of Algorithm 1, for every $j \in \{0, \ldots, \frac{n}{2^i k} - 1\}$ it holds that*

- $q_x^i = q_{x-k}^0$, *for* $x \in \{(2^i j + 1)k, \ldots, 2^i(j+1)k - 1\}$, *and*
- $q_y^i = q_{y+(1-2^i)k}^0$, *for* $y \in \{2^i jk, \ldots, (2^i j + 1)k - 1\}$.

In particular, for $\ell = \log(n) - \log(k)$ it holds $q_z^\ell = q_{z-k}^0$, for $z \in \{0, \ldots, n-1\}$, that is, after ℓ iterations of Algorithm 1 every qubit is correctly placed.

Proof. We prove the lemma by induction on the number of iterations i.

For $i = 1$, it is immediate to see that for every $j \in \{0, \ldots, \frac{n}{2^i k} - 1\}$, every $x \in \{2^i jk, \ldots, (2^i j + 1)k - 1\}$, and every $y = x + 2^{1-1}k = x + k$, the execution of $\text{Swap}(q_x^0, q_y^0)$ yields $q_x^1 = q_y^0 = q_{x-k}^0$ and $q_y^1 = q_x^0 = q_{y-k}^0$.

Let $1 \leq i \leq \log(n) - \log(k)$; assume the claim true for $|q^0\rangle, \ldots, |q^i\rangle$, and let us prove it is true for $|q^{i+1}\rangle$.

By inductive hypothesis, for $j \in \{0, \ldots, \frac{n}{2^i k} - 1\}$, it holds

$$
\begin{aligned}
q_x^i &= q_{x-k}^0, && \text{for } x \in \{(2^i j + 1)k, \ldots, 2^i(j+1)k - 1\}, \text{ and} \\
q_y^i &= q_{y+(1-2^i)k}^0, && \text{for } y \in \{2^i jk, \ldots, (2^i j + 1)k - 1\}.
\end{aligned}
$$

It requires no much effort to check that

$$\{2^{i+1}jk, \ldots, (2^{i+1}j+1)k-1 \mid 0 \leq j \leq \frac{n}{2^{i+1}k} - 1\}$$
$$= \{2^{i}jk, \ldots, (2^{i}j+1)k-1 \mid 0 \leq j \leq \frac{n}{2^{i}k} - 2 \text{ such that } j \text{ is even}\},$$

and that

$$\{(2^{i+1}j+2^{i})k, \ldots, (2^{i+1}j+1+2^{i})k-1 \mid 0 \leq j \leq \frac{n}{2^{i+1}k} - 1\}$$
$$= \{2^{i}jk, \ldots, (2^{i}j+1)k-1 \mid 0 \leq j \leq \frac{n}{2^{i}k} - 1 \text{ such that } j \text{ is odd}\}.$$

It follows that all the qubits involved in a swap during the $(i+1)$st iteration of the algorithm are of the form $q_y^i = q^0_{y-(1-2^i)k}$. Therefore, $q_z^{i+1} = q_z^i$ for $z \in \{(2^ij+1)k, \ldots, 2^i(j+1)k-1\}$ and $j \in \{0, \ldots, \frac{n}{2^ik} - 1\}$.

Moreover, during the $(i+1)$th iteration, the algorithm executes the operation $\texttt{Swap}(q_x^i, q_y^i)$ for $x \in \{2^{i+1}jk, \ldots, (2^{i+1}j+1)k-1\}$ and $y = x + 2^ik$, where $j \in \{0, \ldots, \frac{n}{2^{i+1}k} - 1\}$; such swapping results in

$$q_x^{i+1} = q_y^i = q^i_{x+2^ik} = q^0_{x-(1-2^i)k+2^ik} = q^0_{x-(1-2^{i+1})k}, \text{ and}$$
$$q_y^{i+1} = q^{i+1}_{x+2^ik} = q_x^i = q^0_{x-(1-2^i)k} = q^0_{x+2^ik-k} = q^0_{y-k}.$$

To complete the proof it is enough to observe that

$$\{(2^ij+1)k, \ldots, 2^i(j+1)k-1 \mid 0 \leq j \leq \frac{n}{2^ik} - 1\}$$
$$\cup\{(2^{i+1}j+2^i)k, \ldots, (2^{i+1}j+1+2^i)k-1 \mid 0 \leq j \leq \frac{n}{2^{i+1}k} - 1\}$$
$$=\{(2^{i+1}j+1)k, \ldots, 2^{i+1}(j+1)k-1 \mid 0 \leq j \leq \frac{n}{2^{i+1}k} - 1\}.$$

Indeed, the previous equality implies that

$$q_x^{i+1} = q^0_{x-k}, \text{ for } x \in \{(2^{i+1}j+1)k, \ldots, 2^{i+1}(j+1)k-1\}, \text{ and } j \in \{0, \ldots, \frac{n}{2^{i+1}k} - 1\}.$$

Finally, observe that for $i = \ell = \log(n) - \log(k)$, the variable j assumes only the value 0; furthermore, for $x \in \{0, \ldots, k-1\}$ it holds

$$q_x^\ell = q^0_{x-(1-2^\ell)k} = q^0_{x-k}.$$

Therefore, after $\ell = \log(n) - \log(k)$ iteration of Algorithm 1 every qubit is correctly placed, that is, rightward cyclically shifted by k positions. This completes our proof. □

4. The General Algorithm for $1 \leq k \leq n-1$

In this section we present the algorithm implementing the cyclic rotation operator, shifting the input register of k positions to the right, for the general case in which $1 \leq k \leq n-1$.

ALGORITHM 2: Algorithm for a generic $k \in \{1, \ldots, n-1\}$

Input: a n-qubit register $q := \bigotimes_{x=0}^{n-1} |q_x\rangle$, and $1 \leq k \leq n-1$.
Output: a n-qubit register $q' := \bigotimes_{x=0}^{n-1} |q_{x+k \mod (n)}\rangle$.

1 $\ell := \min\{1 \leq i \leq \log(n) \mid 2^i k = 0 \mod (n)\}$;
2 **for** $i = 1, \ldots, \ell; i++$ **do**
3 **for** $j = 0, \ldots, 2^{\ell - i} - 1; j++$ **do**
4 **for** $x = j\frac{n}{2^\ell}2^i, \ldots, (j2^i + 1)\frac{n}{2^\ell} - 1; x++$ **do**
5 $\mathtt{Swap}(q_x, q_{x+2^{i-1}k \mod (n)})$

The idea behind the general algorithm is very similar to the idea underlying the algorithm for $k = 2^m$. The pseudocode of such general procedure is depicted in Algorithm 2.

Specifically, a new parameter ℓ is defined, by setting

$$\ell := \min\{1 \leq i \leq \log(n) \mid 2^i k = 0 \mod (n)\}.$$

To understand the choice of ℓ, imagine the qubits from $|q\rangle$ arranged circularly. Observe that, by its definition, ℓ is the smallest positive integer such that an interval of length $2^\ell k$ starting at the qubit q_0 finishes at the qubit q_{n-1} when wrapped around $|q\rangle$.

Informally, the main difficulty encountered when trying to extend the approach from Algorithm 1 to the general case consists in the fact that it is not possible to decompose the n-qubits register $|q\rangle$ in disjoint intervals of length $2k$, 2^2k, etc., in general. However, once again, we can imagine that the qubits from $|q\rangle$ are arranged along a circle to which we wrap the decomposition in intervals of length $2^i k$ around. Therefore, we can adapt Algorithm 1 to the general case of an arbitrary k, by reasoning in the n-modular arithmetic. Indeed, by comparing the pseudocodes of the two algorithms, it is immediate to see that in Algorithm 2 we replaced $\log(n) - \log(k)$ by ℓ, and k by $\frac{n}{2^\ell}$.

Figure 4 illustrates the iterations of the algorithm implementing the circular shift operator for a register of 8 qubits in which a rotation of 6 position is performed. Figure 5 and Figure 6 provide an illustration of the application of the circular shift operator as prescribed by Algorithm 2, for a register of 8 qubits in which a rightward circular shift of magnitude 3, 5, and 6 is performed.

Regarding the computational complexity, we observe that - as in the case of Algorithm 1 - in a quantum circuit model of computation the **for**-loops at lines 3 and 4 are executed in parallel. This means that, once ℓ is known, Algorithm 2 executes $\ell \leq \log(n)$ time-steps. Furthermore, to compute the number ℓ of iterations needed, the algorithm has to perform at most $\log(n)$ multiplications. Therefore, the overall time-complexity of Algorithm 2 is $\mathcal{O}(\log(n))$.

Regarding the correctness of the general algorithm, it can be reduced to the following theorem.

Theorem 4.1. *Algorithm 2 correctly outputs the rightward circular shifting of an input n-qubits register q by k positions.*

Proof. Correctness of Algorithm 2 follows from the correctness of Algorithm 1 and in particular from Lemma 3.1. To verify this claim, it is enough to observe that, by the definition of ℓ, in the n-modular arithmetic it holds that $\frac{n}{2^\ell} = k$, $2^{\ell - i}\frac{n}{2^i k}$, and $\log(n) - \log(\frac{n}{2^\ell}) = \ell$. □

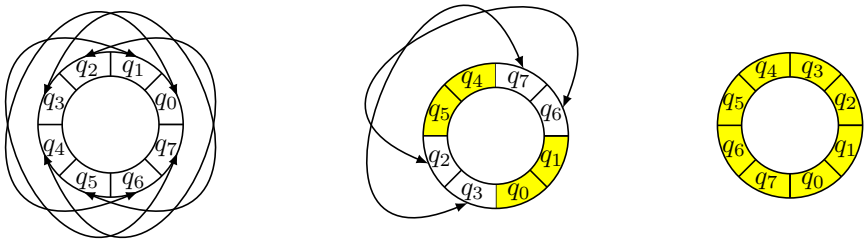

Figure 4: Illustration of the iterations of the algorithm implementing the circular shift operator for a register of 8 qubits in which a rotation of 6 position is performed. The coloured qubits are *placed qubits*. In this circular representation of the register is evident that all swapped pairs have the same distance.

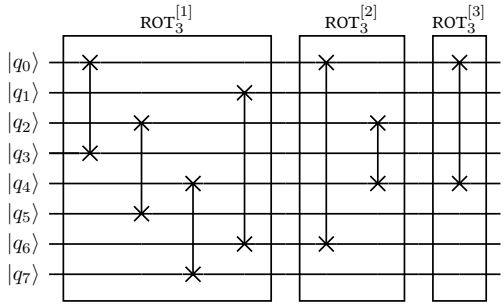

Figure 5: The application of the circular shift operator for a register of 8 qubits in which a rotation of 3 positions is performed. Observe that all the pairs of qubits swapped in the same iteration have the same distance. For instance, in $\text{ROT}_3^{[1]}$, the pair q_6 and q_1 have distance $1 - 6 = 3 \mod (8)$, which is the same distance between the other swapped pairs of qubits. Similarly, all the swapped pairs of qubits in $\text{ROT}_3^{[2]}$ have distance $6 \mod (8)$.

4.1. Adapting the Algorithm to Registers of Any Size

In Section 2, we assumed to work with registers whose length is a power of 2. This does not make our method less general. It is easy to check, indeed, that the following slight modification of Algorithm 2 realizes the circular shift by k position of a qubit-register q of any size. It is enough to replace the register q of arbitrary size n by the register q' that has size $2^{\lceil \log(n) \rceil}$ and such that

$$q'_i = q_i, \text{ for } 0 \leq i < n, \text{ and } q'_i = \star, \text{ for } n \leq i < 2^{\lceil \log(n) \rceil} - 1,$$

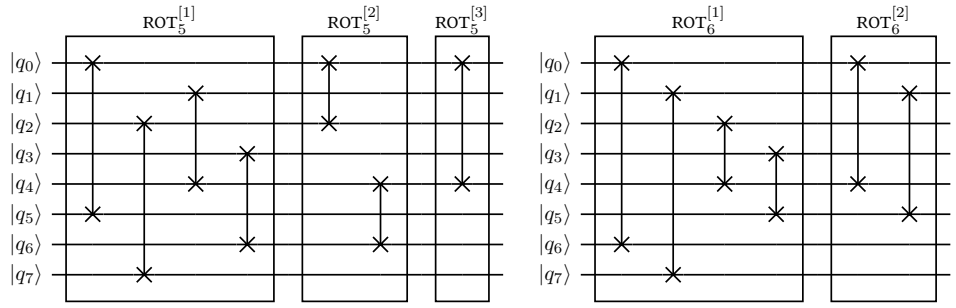

Figure 6: The application of the circular shift operator for a register of 8 qubits in which a rotation of 5 and 6 positions is performed, respectively.

where $\star$ is a special symbol that does not belong to the working alphabet. At this point, it is sufficient to rotate q' by k positions, as dictated by Algorithm 2, and drop from the output register the qubits whose state is equal to $\star$. The resulting register will have size n and be equal to $\bigotimes_{x=0}^{n-1}|q_{x+k \mod (n)}\rangle$, namely the circular shift of q by k positions.

5. Discussion and Conclusions

We have presented a quantum algorithm that performs the rightward circular shift of a quantum register of arbitrary size n by k positions. As we already discussed, the circular shift operator is a staple ingredient of many quantum recipes. For example, in the framework of quantum text processing, it is employed to get all possible portions of a certain fixed length of a text in the [12, 13]. Whereas it was already known that such a gate can be implemented in at most $\log(n)$ steps, a systematic way to build it was missed, and this motivated our work.

Algorithm 2 can be modified to get an algorithm that constructs a quantum gate that implements the rightward circular shift operator $\textsc{rot}_k$, for any $1 \leq k \leq n-1$, in an obvious way, i.e., replacing line 4 of the algorithm by add $\textsc{Swap}(q_x, q_{x+2^{i-1}k \mod (n)})$. We observe that the quantum gate obtained has still depth equal to $\mathcal{O}(\log(n))$. However, there is a non-significant difference between the quantum gate performing ROT and the algorithm that builds it. In fact, during the execution of the quantum gate, there is no need to get the knowledge about the number of needed iterations ℓ (which is already encoded in the circuit implementing the gate), whereas the building algorithm needs to compute it.

We assumed to work with *registers of qubits*, that is, the register is made of 2-dimensional quantum systems. In the quantum computation landscape, this is however not always the case. For example, *qutrits* are quantum systems of dimension 3 that have attracted some interest in quantum cryptography [14, 15]. We think that it makes sense and would be interesting - at least from a purely theoretical perspective - to define the d-dimensional generalisation of the Swap elementary gate and to understand how to use it to implement non-elementary gates performing d-dimensional rotations. To be more concrete, in the framework of qutrits we could define some swap-like elementary gate permuting in some way the states of three qutrits, and then employ it to perform some kind of *spherical shift*.

Acknowledgments

We would like to express our gratitude to Simone Faro for participating in constructive discussions on the problem and for valuable advice. We would also like to thank Domenico Cantone for his suggestions regarding the proof of some lemmas presented in this paper.

References

[1] E. Farhi, J. Goldstone, S. Gutmann, M. Sipser, Quantum computation by adiabatic evolution, 2000. arXiv:quant-ph/0001106.

[2] E. Witten, Topological quantum field theory, Communications in Mathematical Physics 117 (1988) 353 – 386.

[3] R. Jozsa, An introduction to measurement based quantum computation, 2005. arXiv:quant-ph/0508124.

[4] M. A. Nielsen, I. L. Chuang, Quantum Computation and Quantum Information (10th Anniversary edition), Cambridge University Press, 2016. URL: https://www.cambridge.org/de/academic/subjects/physics/quantum-physics-quantum-information-and-quantum-computation/quantum-computation-and-quantum-information-10th-anniversary-edition?format=HB.

[5] A. Zulehner, R. Wille, Simulation and Design of Quantum Circuits, Springer International Publishing, Cham, 2020, pp. 60–82. URL: https://doi.org/10.1007/978-3-030-47361-7_3. doi:10.1007/978-3-030-47361-7_3.

[6] Y. Lao, O. Ait-Aider, Rolling shutter homography and its applications, IEEE Trans. Pattern Anal. Mach. Intell. 43 (2021) 2780–2793. URL: https://doi.org/10.1109/TPAMI.2020.2977644. doi:10.1109/TPAMI.2020.2977644.

[7] D. R. Musser, Introspective sorting and selection algorithms, Softw. Pract. Exp. 27 (1997) 983–993.

[8] R. Weil, J. Vinograd, The cyclic helix and cyclic coil forms of polyoma viral dna, Proceedings of the National Academy of Sciences 50 (1963) 730–738. URL: https://www.pnas.org/doi/abs/10.1073/pnas.50.4.730. doi:10.1073/pnas.50.4.730. arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.50.4.730.

[9] R. Dulbecco, M. Vogt, Evidence for a ring structure of polyoma virus dna, Proceedings of the National Academy of Sciences 50 (1963) 236–243. URL: https://www.pnas.org/doi/abs/10.1073/pnas.50.2.236. doi:10.1073/pnas.50.2.236. arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.50.2.236.

[10] M. Thanbichler, S. Wang, L. Shapiro, The bacterial nucleoid: A highly organized and dynamic structure, Journal of cellular biochemistry 96 (2005) 506–21. doi:10.1002/jcb.20519.

[11] F. Lisacek, Algorithms on strings, trees and sequences: Dan gusfield., Comput. Chem. 24 (2000) 135–137. URL: http://dblp.uni-trier.de/db/journals/candc/candc24.html#Lisacek00.

[12] P. Niroula, Y. Nam, A quantum algorithm for string matching, npj Quantum Information 7 (2021) 37. doi:10.1038/s41534-021-00369-3.

[13] D. Cantone, S. Faro, A. Pavone, Quantum string matching unfolded and extended, in: M. Kutrib, U. Meyer (Eds.), Reversible Computation, Springer Nature Switzerland, Cham, 2023, pp. 117–133.

[14] B. P. Lanyon, T. J. Weinhold, N. K. Langford, J. L. O'Brien, K. J. Resch, A. Gilchrist, A. G. White, Manipulating biphotonic qutrits, Phys. Rev. Lett. 100 (2008) 060504. URL: https://link.aps.org/doi/10.1103/PhysRevLett.100.060504. doi:10.1103/PhysRevLett.100.060504.

[15] M. Smania, A. M. Elhassan, A. Tavakoli, M. Bourennane, Experimental quantum multiparty communication protocols, npj Quantum Information 2 (2016) 16010. URL: https://doi.org/10.1038/npjqi.2016.10. doi:10.1038/npjqi.2016.10.