CONSTRUCTING QUANTUM CIRCUITS FOR SIMPLE PERIODIC FUNCTIONS
FOR QUANTUM INFORMATION AND COMPUTATION

OMAR GAMEL
Department of Physics, University of Toronto, 60 St. George St.,
Toronto, Ontario, M5S 1A7, Canada

DANIEL F. V. JAMES
Department of Physics, University of Toronto, 60 St. George St.,
Toronto, Ontario, M5S 1A7, Canada

Received (May 9, 2013)
Revised (revised date)

Periodic functions are of special importance in quantum computing, particularly in applications of Shor’s algorithm. We explore methods of creating circuits for periodic functions to better understand their properties. We introduce a method for constructing the circuit for the simplest periodic function, that is one-to-one within a single period, of a given period $p$. We conjecture that to create the simplest periodic function of period $p$, where $p$ is an $n$-bit number, one needs at most $n$ Toffoli gates.

Keywords: Quantum Circuit, Periodic Function, Shor’s Algorithm

Communicated by: to be filled by the Editorial

1 Introduction

As quantum computers improve and grow in size beyond a dozen or so qubits, they face two daunting problems. Firstly, since complete tomographic reconstruction of the quantum state becomes increasing intractable, how might these devices be characterized and their performance validated? Secondly, and by no means distinctly, is the problem of finding meaningful milestones for device development along the long road to true large scale devices capable of tackling useful problems.

Shor’s factoring algorithm, nearly 20 years since it was invented, remains arguably the most promising and compelling application of quantum computing. It allows one to factor large numbers in polynomial time, undermining the most common cryptographic schemes in use today, such as RSA cryptography. The well known algorithm is based on the quantum Fourier transform to find the period of a function. Highly simplified versions of this algorithm have been implemented using both NMR and linear optical implementations, providing just such a milestone for two of the possible technologies under consideration as quantum computers.

Shor’s algorithm make use of the periodicity of the modular exponential function, which can be evaluated efficiently due to the inherent massive parallelism of quantum computing. The quantum Fourier transform can then be used to extract the period of the function, from which...
desired factors can be deduced. The key “quantum” elements are the parallel evaluation of a periodic function and the extraction of its period. This motivates us to better understand periodic functions as an independent object of study in quantum computing. In particular, it is of interest to study how a quantum circuit can be created to implement functions of a given period, and the resources such a circuit requires. Moreover, given the limited capacity of currently realizable experimental systems, it is of interest to find the minimal number of gates needed for a function of a given period. This will provide milestones as experimental technology advances.

In this paper, we investigate the process of creating a quantum circuit for the simplest periodic function of given period $p$, using only CNOT and Toffoli gates. We begin by defining what we mean by the simplest periodic function. Then we give illustrative step by step examples for constructing some circuits of the simplest periodic function for some values of period $p$, with the minimum possible number of quantum gates. Finally, we conjecture an upper limit to the required number of Toffoli gates in such a circuit for any period $p$.

2 Periodic Functions

We define a periodic injective function as any function $F_{p,n}$ satisfying the following properties: it is a binary function with $n$ input bits whose output has a period $p$, and is one-to-one (i.e. injective) within a single period. That is, the function has $2^n$ possible input values, so the argument of the function, $x$, is a binary number between 0 and $2^n - 1$, inclusive. We require that $y = F_{p,n}(x)$ take on unique values (i.e. is one-to-one) for $0 \leq x < p$, and that $F_{p,n}(x) = F_{p,n}(x - p)$ for $x \geq p$.

Writing the input $x$ in binary notation, we have $x = x_n...x_2x_1$, where the $x_i$ denote each of the $n$ input bits. For example, if $x$ has the value 13 in decimal notation, then in binary notation it is $x = 1101$, with individually bits $x_4 = x_3 = x_1 = 1$ and $x_2 = 0$. Similarly, if $y$ is the output, we write it in binary notation as $y = y_m...y_2y_1$ and the $y_j$ denote each of the $m$ output bits. Note that $m$ need only be large enough so the number $p$ can be represented in $m$ bits, i.e. $m = \lceil \log_2(p) \rceil$, where $\lceil w \rceil$ is the ceiling of $w$ (defined as the smallest integer which is not smaller than $w$; for example $\lceil 3.142 \rceil = 4$, $\lceil 5 \rceil = 5$).

We further define a simple periodic function $G_p$ as a periodic injective function with period $p$, and a number of bits enough to contain just one complete period $p$. That is, $n = m = \lceil \log_2(p) \rceil$. So we write,

$$G_p \equiv F_{p,\lceil \log_2(p) \rceil}.$$  

(2.1)

Note that there are many different simple periodic functions $G_p$ for a given $p$. In what follows, we seek to construct the quantum circuit for the function $G_p$ with the smallest Toffoli gate count. This is since Toffoli gates are the most demanding to implement experimentally, and minimizing them makes the construction easier. We call the $G_p$ with the smallest Toffoli gate count $S_p$ (which in general is not a unique function).

3 Constructing Circuits

We now address the task of constructing the quantum circuit for $S_p$, for a given $p$. In fact, it is by constructing the circuit for a function $G_p$ while trying to minimize the number of Toffoli gates that $S_p$ can be found. Below, we give examples of the function table and the
quantum circuit for different values of $p$. Note that if $p$ is even, then the quantum circuit for $S_p$ is simply the circuit of $S_2^p$ constructed between the input and output bits $x_i$ and $y_i$ for $(i = 2, ..., n)$, with an additional controlled-not gate that copies $x_1$ to $y_1$. With this in mind, we are only interested in odd $p$.

The circuits below are constructed by inspection, and trial and error, making use of some general patterns and principles. Each circuit can be seen as two processes (which can occur in tandem). The first is copying a linear combination of the input qubits to each output qubit via CNOT gates. This process must be used to create linearly independent combinations of the input bits in output bits, which serve as the canvas on which the second process will operate. The second process is the application of cascades of Toffoli gates to modify the results of the first process by flipping some entries in the function table. It is this second process where most of the creativity lies, since it is what actually creates the periodicity. Note that the two processes may be made to occur in tandem.

If one thinks of the function tables below, the first Toffoli gate applied modifies a number of entries equal to a quarter of the total length of the column (i.e. $2^{n-2}$). Each consecutive Toffoli gate in the cascade flips half the number of entries of its target qubit as in the previous level of the cascade. For example, if $n = 4$, then the first Toffoli gate will flip $2^{4-2} = 4$ entries, the second gate in the cascade will flip 2 entries, and the third gate will flip 1 entry.

Where possible, the output of a Toffoli cascade is copied over to other qubits via CNOT gates, to avoid duplication of effort. During the circuit construction, we try to minimize the implementation cost of the circuit. Toffoli gates are most difficult, they take about six CNOT gates to implement, in addition to some comparatively cheap single qubit gates. Therefore one must minimize the gate count, roughly with six CNOT gates equal to one Toffoli gate.

We start by constructing the function table and circuit for $S_3$, for the first few values of odd $p$. We first consider $S_3$, which implies $n = 2$ bits each for the input and output registers. We see below that we will need 1 Toffoli gate, and 3 CNOT gates. The process is as follows, we set $y_2 = x_1 \oplus x_2$, then we use a Toffoli gate with $y_1$ as its target, which we then copy onto $y_2$. We use standard notation for CNOT and Toffoli gates, with a black circle indicating the control qubit, and a large circle with a plus sign inside indicates the target qubit. The small white circles are inverted control qubits, in the sense that the target bit is modified if the inverted control bit has value 0, and is unchanged if it has value 1.

The stars in the function table denote the entries flipped by the action of a Toffoli gate, whether directly or indirectly (where the result of the Toffoli is copied by a CNOT to another bit). Each star is an entry flip, so an even number of stars leaves the entry unchanged. The single horizontal line indicates where the second period begins, and the values of the output bits under this line must repeat the values above it.
Quantum circuits for simple periodic functions...

Similarly, the following are the circuit and table for \( S_5 \), which requires need 2 Toffoli gates, and 3 CNOTs. It involves setting \( y_1 = x_1 \oplus x_3 \) and \( y_2 = x_2 \), then using a cascade of two Toffoli gates to modify three entries in the function table. The modified entries are once again marked with a star. Note that value of \( y_3 \) was not created by using CNOT additions from the \( x_i \), rather it was 'synthetized' through cascades of Toffoli gates.

The function \( S_7 \), needs 2 Toffoli gates, 4 CNOTs. Here we set \( y_1 = x_1 \) and \( y_2 = x_2 \), then we use a cascade of two Toffoli gates to modify three entries in the function table, followed by a CNOT which duplicates one of these modified entries to another output bit, bringing the total number of modified entries to four, which are marked with a star in the table. Finally, \( x_3 \) is added to \( y_3 \). Note that this final step was not done earlier in the process to facilitate the copying of the result of the second Toffoli to another bit.

The function \( S_9 \), needs 3 Toffoli gates, 4 CNOTs. In a process very similar to the \( S_5 \) circuit, we set \( y_1 = x_1 \oplus x_4 \), \( y_2 = x_2 \), and \( y_3 = x_3 \). Then a cascade of three Toffoli gates is used to modify seven entries in the function table. Once again, the value of \( y_4 \) is synthesized solely using Toffoli gates.

Note that the circuit for \( S_5 \) and \( S_9 \) have the same structure. In fact, the pattern for these circuits can be generalized for any \( S_p \) where \( p = 2^k + 1 \) for some positive integer \( k \). Similarly, the pattern in \( S_7 \) can be generalized to \( S_{15} \) and any \( S_p \) where \( p = 2^k - 1 \) for some positive
integer $k$.

The function $S_{11}$, needs 4 Toffoli gates, 5 CNOTs. Here the construction of the circuit is more complicated. We start by setting $y_1 = x_1 \oplus x_4$ and $y_2 = x_2$. We then follow it by a cascade of three Toffoli gates, and then we add $x_3 \oplus x_4$ to $y_3$, which again we chose to do after the cascade of Toffoli gate. Finally, we add a fourth Toffoli gate to synthesize the contents of $y_4$. Overall, nine entries in the function table have been modified by Toffoli gates, since an entry with an even number of stars means its bit value was flipped twice, and therefore it was unchanged.

4 Larger Circuits

One can construct these circuits for arbitrary odd numbers. We have continued this process for larger circuits, and the following table gives the number of Toffoli and CNOT gates for the circuit of the function $S_p$ for all odd periods $p$ up to 5 bits. Note that $p_2$ is the period $p$ expressed in base 2.

\begin{table}[h]
\centering
\begin{tabular}{cccc|cccc}
\hline
$x_4$ & $x_3$ & $x_2$ & $x_1$ & $y_4$ & $y_3$ & $y_2$ & $y_1$ \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
\end{tabular}
\end{table}

Fig. 4. The quantum circuit and function table for the function $S_9$.

Given the above information, one can conjecture the following main result of this introductory paper: to create the simplest periodic function of period $p$, where $p$ is an $n$-bit number, one needs at most $n$ Toffoli gates. More precisely, let $c_2$ be the binary string equal to $p_2$ with the last bit truncated (since it is always 1, because $p$ is odd). Then, we conjecture that for a given $p$, if the respective $c_2$ contains the substring 01, which we call type 1, then exactly $n$ Toffoli gates are needed for the simplest periodic function. If the substring 01 does not occur in $c_2$, which we call type 2, then exactly $n - 1$ Toffoli gates are needed.

For a given bit length $n$, there are $2^n - 2$ odd periods $p$ where $p$ is $n$-bits. Of these $2^n - 2$
Quantum circuits for simple periodic functions

Fig. 5. The quantum circuit and function table for the function $S_{11}$.

Table 1. Number of Toffoli and CNOT gates needed to construct the circuit for $S_p$

| Period $p$ | Period $p_2$ | Bitlength $n$ | Toffoli $n$ | CNOT $n$ |
|------------|--------------|---------------|-------------|-----------|
| 3          | 11           | 2             | 1           | 3         |
| 5          | 101          | 3             | 2           | 3         |
| 7          | 111          | 3             | 2           | 4         |
| 9          | 1001         | 4             | 3           | 4         |
| 11         | 1011         | 4             | 4           | 5         |
| 13         | 1101         | 4             | 3           | 6         |
| 15         | 1111         | 4             | 3           | 5         |
| 17         | 10001        | 5             | 4           | 5         |
| 19         | 10011        | 5             | 5           | 6         |
| 21         | 10101        | 5             | 5           | 6         |
| 23         | 10111        | 5             | 5           | 7         |
| 25         | 11001        | 5             | 4           | 8         |
| 27         | 11011        | 5             | 5           | 7         |
| 29         | 11101        | 5             | 4           | 7         |
| 31         | 11111        | 5             | 4           | 6         |
possible odd periods, \( n - 1 \) will be of type 2, and the rest of type 1.

As an example, for \( p = 23 \), we have \( p_2 = 10111 \), and \( c_2 = 1011 \), which does contain the substring 01, i.e. is type 1, therefore \( n = 5 \) Toffoli gates are needed. In the case \( p = 25 \), then \( p_2 = 11001 \), and \( c_2 = 1100 \), which does not contain the substring 01, therefore it is type 2 and \( n - 1 = 4 \) Toffoli gates are needed. A proof of this conjecture will be addressed in a future work.

Note that this conjecture also be simplified by stating the following more useful statement: to create the circuit for the simplest periodic function \( S_p \), where the period \( p \) is an \( n \)-bit number, one needs at most \( n \) Toffoli gates.

5 Conclusion

We have demonstrated a custom procedure for constructing the circuit for the simplest periodic function \( S_p \) using CNOT and Toffoli gates. The procedure is easily scaleable for periods \( p \) of special forms \( p = 2^k \pm 1 \), and otherwise can be scaled on a custom bases. We conjecture that for \( p \) an \( n \)-bit number, one needs at most \( n \) Toffoli gates to construct \( S_p \). These simple periodic circits may serve as stepping stones for experimental procedures as technology improves.

In future work, we will address a scaleable procedure for more general forms of \( p \), and provide a proof for the conjecture above. Moreover, we will analyze the periodic properties of these functions, by examining how they behave under Fourier transforms. Finally, we will generalize the problem at hand to more complicated periodic functions with more than just one complete period.

References

1. M. Kaznady and D. F. V. James, Numerical strategies for quantum tomography: Alternatives to full optimization, Phys. Rev. A 79, 022109, (2009).
2. P. Shor (1997), Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, SIAM Journal of Computing 26, pp. 1484-1509.
3. R. Rivest, A. Shamir and L. Adleman (1978), A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM 21 (2), 120-126.
4. M. Nielsen and I. Chuang (2000), Quantum Computation and Quantum Information, Cambridge University Press.
5. L. Vandersypen et. al, Experimental realization of Shor’s quantum factoring algorithm using nuclear magnetic resonance, Nature 414 (6866), 883-887, (2001).
6. Lanyon et. al, Experimental Demonstration of a Compiled Version of Shor’s Algorithm with Quantum Entanglement, Phys. Rev. Lett. 99, 250505 (2007).
7. Le et. al, Demonstration of a Compiled Version of Shors Quantum Factoring Algorithm Using Photonic Qubits, Phys. Rev. Lett. 99, 250504 (2007).
8. Martin-Lopez et. al, Experimental realization of Shor’s quantum factoring algorithm using qubit recycling, Nature Photonics 6, 773-776 (2012).
9. Lanyon et. al, Experimental Demonstration of a Compiled Version of Shor’s Algorithm with Quantum Entanglement, Phys. Rev. Lett. 99, 250505 (2007).