OPEN ACCESS Freely available online

PLoS one

Interfacing External Quantum Devices to a Universal Quantum Computer

Antonio A. Lagana[1]*, Max A. Lohe[2], Lorenz von Smekal[3]

1 School of Chemistry and Physics, University of Adelaide, Adelaide, South Australia, Australia, **2** School of Chemistry and Physics, University of Adelaide, Adelaide, South Australia, Australia, **3** Institut für Kernphysik, Technische Universität Darmstadt, Darmstadt, Germany

Abstract

We present a scheme to use external quantum devices using the universal quantum computer previously constructed. We thereby show how the universal quantum computer can utilize networked quantum information resources to carry out local computations. Such information may come from specialized quantum devices or even from remote universal quantum computers. We show how to accomplish this by devising universal quantum computer programs that implement well known oracle based quantum algorithms, namely the Deutsch, Deutsch-Jozsa, and the Grover algorithms using external black-box quantum oracle devices. In the process, we demonstrate a method to map existing quantum algorithms onto the universal quantum computer.

Citation: Lagana AA, Lohe MA, von Smekal L (2011) Interfacing External Quantum Devices to a Universal Quantum Computer. PLoS ONE 6(12): e29417. doi:10.1371/journal.pone.0029417

Editor: Gerardo Adesso, University of Nottingham, United Kingdom

Received November 1, 2011; **Accepted** November 28, 2011; **Published** December 28, 2011

Funding: The authors have no funding or support to report.

Competing Interests: The authors have declared that no competing interests exist.

* E-mail: antonio.lagana@adelaide.edu.au

Introduction

Quantum networks which connect quantum systems and can transmit quantum information have been extensively discussed [1]. Quantum connectivity provides a means of overcoming size-scaling and error-correction problems, and has significant advantages over classical connectivity. Furthermore, networks of quantum computers have also been proposed [2] where information can be exchanged between nodes via quantum and classical channels. A general question arises as to whether and how such quantum computers can communicate and exchange information. In the simplest case a quantum computer may download data sets from other nodes over the quantum network, but in more complex cases use the network to call subroutines, or concatenate programs from other quantum computers.

It is well known that classical principles do not necessarily apply in the realm of quantum mechanics. The no-cloning theorem (see [3] for example) is a well-known example of this. In the field of quantum computing, the ability to halt a programmable quantum computer was such an example. The original Universal Quantum Turing Machine proposal [4] made the tacit assumption that a quantum turing machine could be halted in a classical manner. This turned out to be problematic (see [5] for a discussion of the issues associated with the original proposal) due to properties of quantum mechanics. Thus, it is imperative to formally show whether a classical solution or property is applicable (or even relevant) in the realm of quantum mechanics. Assuming that a classical solution to a problem directly applies to a quantum mechanical system is prone to run into potential complications.

We address here the question of how a universal quantum computer can access an external oracle, which may be regarded as a "black box" quantum device, possibly over a quantum network but in any case as a separate and external quantum system to the universal quantum computer itself. In fact, the oracle may be a program running on a remote universal quantum computer. It should be noted that this is a different problem from that of implementing an oracle "program" on a universal quantum computer. This is of course possible by virtue of the fact that the computer is universal. Hence, if such a program exists, it can be implemented and executed on a universal quantum computer. Strictly speaking, however, the ability to utilize external quantum devices over a network connection is a different problem because such devices are external to the universal quantum computer itself.

Classically, the ability to access devices on a network is a well-known problem with well-known solutions. However, as stated earlier, we cannot assume that this is necessarily the case for a quantum computer accessing quantum devices on a quantum network. Our aim is to explicitly show that accessing external quantum devices with a universal quantum computer is indeed possible by devising universal quantum computer programs that implement well-known oracle based quantum algorithms, namely the Deutsch, Deutsch-Jozsa, and the Grover algorithms using external black-box quantum oracle devices.

In [5] we constructed a programmable universal quantum computer *UQC* that is universal in the sense that it can emulate any classical Turing machine and can approximate any unitary operation to any desired accuracy. It is programmable in the sense that the machine's operations are specified using a sequence of instructions in the same way as for classical computers. *UQC* also supports conditional branching and hence conditional execution, a feature that is not directly possible in the quantum gate array circuit framework. Moreover, *UQC* uses a halting scheme that allows for valid program concatenation, thus resolving issues with the original Universal Quantum Turing Machine (UQTM) proposed by Deutsch [4].

In order to use information from a quantum network in *UQC* programs, we need to devise a means of enabling *UQC* programs to access such remote information and use that information for local computations. We assume that remote quantum nodes exist and treat them as black boxes without any assumptions as to their internal structure or operational details. Without loss of generality, we assume that such devices accept a finite number of input qubits and generate a finite number of output qubits. The input and output qubits may be shared, which is the case if the remote device functions in such a way as to alter the input qubits based on its function. We also assume, without loss of generality, that quantum network nodes have an "enable" qubit, $|\mathrm{en}\rangle$, that controls when an access is to begin, in order to let the device know when the input data has been prepared and is valid. We further assume, without loss of generality, that the nodes of the network generate their output data in less time than the time associated with a single iteration of *UQC*. If the query time were longer than a single iteration of *UQC* or were data-dependent, one could simply write the *UQC* program to wait for the appropriate number of cycles before using the result of the network access. Alternatively, the nodes could provide an "access completed" status flag qubit such that the *UQC* program could poll this status flag qubit before using the result of a network access.

Results

Recall from [5] that *UQC* consists of a memory tape M with an infinite number of qubits, of which only a finite portion is ever used, and a processor that contains observables that play the roles of several registers, including a data register D, a program counter register P, a scratch qubit s, and the halt qubit h. The processor executes programs stored on the memory tape using data that is also stored on the memory tape. A program of *UQC* consists of a sequence of qubits whose states encode instructions of the instruction set defined in [5] and reproduced in Table 1 at the end of this paper.

The single qubit operations **H** and **T** act on the qubit at tape location $M(D)$, denoted $|M\rangle_D$, and the two qubit operations **SWAP** and **NAND** act on $|M\rangle_D$ and the scratch qubit $|s\rangle$, the latter being used as the control qubit for the **NAND** operation.

The instruction set includes a set of operations that can approximate any unitary operation to any desired accuracy. Thus, it is quantum computationally universal. In [5] we constructed a *UQC* program that can compute the **NAND** function, thereby showing that the machine can compute any classically computable function. Because of *UQC*'s universality, any algorithm that can be implemented in the quantum gate array framework can be mapped to an equivalent *UQC* program by virtue of the fact that gate array circuits can be decomposed into circuits of gates with the same universal set of unitary operations {H,T,CNOT} that are implemented in the *UQC* instruction set. Each of the qubits in a quantum circuit (i.e. lines connecting gates) can be mapped to a suitable memory tape data qubit and each of the unitary operations (i.e. quantum gates) can be mapped to a suitable *UQC* subroutine. It is possible therefore to map quantum gate array implementations of algorithms such as the quantum Fourier transform, quantum phase estimation, quantum order finding, quantum factoring discussed in [6] (Chapter 5) onto *UQC*.

Table 1. *UQC* Instruction Set.

Label	Encoding	Description
$\|\mathrm{NOP}\rangle$	$\|0000\rangle$	No operation
$\|D\rightarrow 0\rangle$	$\|0001\rangle$	$D\rightarrow 0$
$\|D+1\rangle$	$\|0010\rangle$	$D\rightarrow D+1$
$\|D-1\rangle$	$\|0011\rangle$	$D\rightarrow D-1$
$\|\mathrm{H}\rangle$	$\|0100\rangle$	Apply Hadamard operation to $\|M\rangle_D$
$\|\mathrm{T}\rangle$	$\|0101\rangle$	Apply $\pi/8$ operation to $\|M\rangle_D$
$\|\mathrm{SWAP}\rangle$	$\|0110\rangle$	$\|M\rangle_D \leftrightarrow \|s\rangle$
$\|\mathrm{CNOT}\rangle$	$\|0111\rangle$	CNOT of $\|M\rangle_D$ and $\|s\rangle$ ($\|s\rangle$: control)
$\|D\leftrightarrow P\rangle$	$\|1000\rangle$	$\|D\rangle \leftrightarrow \|P\rangle$ (branch) iff $s=0$
$\|\mathrm{CLS}\rangle$	$\|1001\rangle$	Clear s
$\|h\rightarrow 1\rangle$	$\|1111\rangle$	$\|h\rangle \rightarrow \|1\rangle$ (set halt qubit)

doi:10.1371/journal.pone.0029417.t001

Accessing Networked Quantum Resources With *UQC*

Modifying *UQC* to use networked quantum devices, then, is a matter of connecting the qubits comprising the interface (input, output, enable, and optionally access complete) qubits of those devices to a finite subset of the data portion of M, which is the quantum analog of a classical computer's memory-mapped I/O and allows *UQC* programs to access remote devices using the M qubits that are connected to those devices. The *UQC* programs prepare the appropriate input data qubits, set the corresponding access enable qubits to perform an access, and utilize the corresponding output data qubits of M. It should be noted that a remote quantum device could be another instance of *UQC* which would enable distributed quantum computing. However, the scheme to access data from remote devices, be they simple devices or full-fledged quantum computers, would work in the same way.

Primitive Programs

In [5] we defined several primitive programs and subroutines that serve as building blocks for devising and analyzing more complicated and useful programs. We reproduce here only those that we specifically require for constructing the algorithms that are the focus of this work. By considering the quantum gate array framework implementations of the algorithms, we identify that we need programs that perform the operations **H**, σ_x, and **CNOT**. We also need to swap qubits for several operations such as enabling or disabling the remote networked quantum device, and the ability to address individual qubits on the memory tape to perform operations on them. Finally, we need a primitive program to halt the overall program.

In the equations that follow, superscripts on programs denote the operation specified by the program and subscripts indicate the qubits on which the program specifies the processor to operate upon. For notational simplicity, $|P_h\rangle$ denotes the program that halts *UQC*, i.e. $|P_h\rangle \stackrel{\mathrm{def}}{=} |h\rightarrow 1\rangle|\mathrm{NOP}\rangle$.

The first set of primitive programs, $\{|D_{+i}\rangle, |D_i\rangle, |S_{i,s}\rangle, |S_{i,j}\rangle\}$, is a subset of those defined in [5]:

1. $|D_{+i}\rangle$: Increment D by i,

$$|D_{+i}\rangle \stackrel{\mathrm{def}}{=} \begin{cases} \Pi_{k=1}^{i}|D+1\rangle & \text{if } i \geq 1, \\ \mathbb{I} & \text{otherwise.} \end{cases} \quad (1)$$

2. $|D_i\rangle$: Set D to i, $i>0$,

$$|D_i\rangle \stackrel{\mathrm{def}}{=} |D+1\rangle|D\rightarrow 0\rangle|D_{+i}\rangle. \quad (2)$$

Recall from the discussion of U_{EX} in [5], that we precede the $D\rightarrow 0$ instruction with a $D+1$ instruction to ensure that $D>0$ when the $D\rightarrow 0$ instruction is executed.

3. $|S_{i,s}\rangle$: Swap data qubits $D(i)$ and s,

$$|S_{i,s}\rangle \stackrel{\text{def}}{=} |D_{5i-1}\rangle|\text{SWAP}\rangle. \tag{3}$$

4. $|S_{i,j}\rangle$: Swap data qubits $D(i)$ and $D(j)$,

$$|S_{i,j}\rangle \stackrel{\text{def}}{=} |S_{5i-1,s}\rangle|S_{5j-1,s}\rangle|S_{5i-1,s}\rangle. \tag{4}$$

We also describe the set of programs $\{|P_i^{\text{H}}\rangle,|P_{i,j}^{\text{H}}\rangle,|P_{i,j}^{\text{C}}\rangle\}$ which apply the single- and multiple-qubit H and CNOT operations on arbitrary qubits on the memory tape, where i and $j\in\mathbb{Z}$:

1. $|P_i^{\text{H}}\rangle$: Apply H to data qubit $D(i)$,

$$|P_i^{\text{H}}\rangle \stackrel{\text{def}}{=} |D_{5i-1}\rangle|\text{H}\rangle. \tag{5}$$

2. $|P_{i,j}^{\text{H}}\rangle$: Apply H to data qubits $D(i:j)$, where $i\geq j$,

$$|P_{i,j}^{\text{H}}\rangle \stackrel{\text{def}}{=} \Pi_{k=j}^{i}|P_k^{\text{H}}\rangle. \tag{6}$$

One could implement this program using a loop but that would require first implementing binary addition of M qubits. Binary addition is possible because one can implement a binary adder such as a Carry Lookahead Adder (CLA) [7] using the NAND program that we defined in [5]. However, since we are only interested in a polynomial order (in the number of qubits) multiple qubit Hadamard transformation program, we define $|P_{i,j}^{\text{H}}\rangle$ as a sequential "unrolled" loop program.

3. $|P_{i,j}^{\text{C}}\rangle$: Apply CNOT to data qubits $D(i)$ and $D(j)$ with $D(i)$ as the control qubit,

$$|P_{i,j}^{\text{C}}\rangle \stackrel{\text{def}}{=} |S_{i,s}\rangle|D_{5j-1}\rangle|\text{CNOT}\rangle|S_{i,s}\rangle. \tag{7}$$

Using the primitive programs defined above, we define $|P_i^{\text{X}}\rangle$ as the program that applies the σ_x (X) operation on data qubit $i\in\mathbb{Z}^+$. Noting that a CNOT operation with the control qubit in the $|1\rangle$ state is equivalent to the X operation, we deduce the equivalence

$$|P_i^{\text{X}}\rangle \equiv |P_{|1\rangle,i}^{\text{C}}\rangle, \tag{8}$$

where the subscript $|1\rangle$ denotes that some suitable data qubit on the memory tape has been prepared in the state $|1\rangle$. Similarly, we define $|P_i^{\text{Z}}\rangle$ as the program that applies the σ_z (Z) operation on data qubit $i\in\mathbb{Z}^+$. Noting that HXH = Z, we deduce

$$|P_i^{\text{Z}}\rangle \equiv |P_i^{\text{H}}\rangle|P_i^{\text{X}}\rangle|P_i^{\text{H}}\rangle. \tag{9}$$

Finally, we define a program $|P_{i,j}^{\text{CZ}}\rangle$ that conditionally applies the Z operation on data qubit $i\in\mathbb{Z}^+$ and data qubit $j\in\mathbb{Z}^+$. Since CNOT is the conditional X operation, we have

$$|P_{i,j}^{\text{CZ}}\rangle \equiv |P_j^{\text{H}}\rangle|P_{i,j}^{\text{C}}\rangle|P_j^{\text{H}}\rangle. \tag{10}$$

UQC Algorithms Using Networked Quantum Oracle Devices

With the notable exception of Shor's factorization algorithm [8], several well known quantum algorithms that achieve a speed-up over their fastest known classical counterparts rely on the use of an oracle, the best known examples being the Deutsch, Deutsch-Jozsa, and Grover algorithms (see Nielsen and Chuang [6], for example). The Deutsch algorithm can determine a global property of a function $f(x)$, namely $f(0)\oplus f(1)$, using only one evaluation of $f(x)$ whereas the fastest classical algorithm requires at least two evaluations of $f(x)$. The Deutsch-Jozsa algorithm can determine whether a two-valued (0 or 1) function $f(x)$ is constant or balanced with only one evaluation of $f(x)$ whereas the fastest classical algorithm requires $2^{n-1}+1$ evaluations, where n denotes the number of bits required to encode the possible values of $f(x)$. Grover's algorithm [9] can find a marked item in an unstructured database of N elements in $O(\sqrt{N})$ operations whereas the fastest classical algorithm requires $O(N)$ operations. Thus, these quantum algorithms all achieve at least a quadratic speedup over their classical counterparts.

These algorithms are well suited to illustrate the use of networked quantum resources with the *UQC* because they rely on black-box quantum devices that generate some output based on the given input. They thus serve as prototypical examples of a networked quantum node, whose internal implementation details are unknown; only the interface protocol need be known. Here, we assume the simplest protocol, which is that the output is valid one "clock cycle" after making a request.

Deutsch and Deutsch-Jozsa Algorithms on *UQC*. We now illustrate the use of a networked quantum device in a *UQC* program by first implementing the simplest known oracle based quantum algorithm, Deutsch's algorithm. The Deutsch oracle works as follows:

$$|x,y\rangle\rightarrow\begin{cases}|x,y\oplus f(x)\rangle & \textit{if } |\text{en}\rangle = |1\rangle\,,\\ |x,y\rangle & \textit{otherwise},\end{cases}$$

where f is some function and $|\text{en}\rangle$ denotes the oracle query enable flag. The memory tape is prepared with $D(0)=|0\rangle$ and $D(1)=|1\rangle$ where $D(0)$ and $D(1)$ take the roles of x and y, respectively. We assume without loss of generality that $D(2)$ takes the role of $|\text{en}\rangle$ and is prepared as $|0\rangle$, and $D(3)$ is initially prepared as $|1\rangle$.

The program that executes the Deutsch algorithm is

$$|P_D\rangle \stackrel{\text{def}}{=} |P_{1,0}^{\text{H}}\rangle|S_{2,3}\rangle|S_{2,3}\rangle|P_0^{\text{H}}\rangle|P_h\rangle, \tag{11}$$

where $|P_{1,0}^{\text{H}}\rangle$ applies the Hadamard transform to the data qubits corresponding to x and y. $|S_{2,3}\rangle|S_{2,3}\rangle$ swap qubits $D(2)$ and $D(3)$ thereby setting the oracle's $|\text{en}\rangle$ qubit (recall that $D(2)$ is connected to $|\text{en}\rangle$ and that $D(2)=|0\rangle$ and $D(3)=|1\rangle$ initially) for a single *UQC* cycle and then clears it, returning the state of $D(3:2)$ back to the original state. At this point, the oracle has generated the output state $|D(0),D(0)\oplus D(1)\rangle$. $|P_0^{\text{H}}\rangle$ then applies the Hadamard transform to the x output of the oracle and $|P_h\rangle$ halts the program thus yielding the following on the memory tape:

$$|D(0),D(1)\rangle = \pm|f(0)\oplus f(1)\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right].$$

Measuring $D(0)$ yields the result that we were interested in, $f(0)\oplus f(1)$. This is a specific mapping of the gate array implementation of the algorithm (see [6] Figure 1.19, for example) onto the instruction set of *UQC*.

We can similarly implement the Deutsch-Jozsa algorithm by mapping a gate array implementation such as the one shown in [6], Figure 1.20. In this case, data qubits $D(0:n-1)$ take the role of x, $D(n)$ takes the role of y, and we use $D(n+1)$ as the $|\mathrm{en}\rangle$ qubit. As before, $D(0:n-1)$ are prepared in the $|0\rangle$ state, $D(n)$ is prepared in the $|1\rangle$ state, $D(n+1)$ is prepared in the $|0\rangle$ state and $D(n+2)$ is prepared in the $|1\rangle$ state. The Deutsch-Jozsa oracle works like the Deutsch oracle with the only difference being that x is n qubits wide. The resulting *UQC* program that computes the Deutsch-Jozsa algorithm is therefore

$$|P_{DJ}\rangle \stackrel{\mathrm{def}}{=} |P^{\mathrm{H}}_{n-1,0}\rangle|S_{n+1,n+2}\rangle|S_{n+1,n+2}\rangle|P^{\mathrm{H}}_{n-1,0}\rangle|P_h\rangle, \quad (12)$$

which is again a direct mapping of the gate array implementation onto the *UQC* instruction set.

Grover's Algorithm on *UQC*. We now use the techniques developed in the previous section to implement the Grover unstructured database search algorithm. We assume that the database has only one marked solution as can be determined by using the quantum counting algorithm (see [6] Chapter 6, for example). We denote the query data qubits as $|q\rangle$ and the query enable flag as $|\mathrm{en}\rangle$. The Grover oracle works as follows:

$$|q\rangle \rightarrow \begin{cases} (-1)^{f(q)}|q\rangle & \textit{if } |\mathrm{en}\rangle = |1\rangle, \\ |q\rangle & \textit{otherwise} \end{cases}$$

where $f(q)=1$ if q is a solution to the search problem and $f(q)=0$ otherwise. More concisely, the oracle performs the unitary transformation

$$U_m \stackrel{\mathrm{def}}{=} \mathbb{I} - 2|m\rangle\langle m|, \quad (13)$$

where $|m\rangle$ denotes the marked solution. In other words, the oracle flips the phase of the solution state but leaves non-solution states unchanged. Grover's algorithm prepares an initial query state as the equal superposition of all elements in the database, followed by $O(\sqrt{2^n})$ iterations of G, where

$$G \stackrel{\mathrm{def}}{=} (2|s\rangle\langle s| - \mathbb{I})U_m, \quad (14)$$

and

$$|s\rangle = \frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}|i\rangle \quad (15)$$

denotes the equal superposition of all database elements.

Thus, the first step in the program is to create a superposition of all database items in $D(n:1)$ where $D(i)=M(5i-1)$, $i\in\mathbb{Z}^+$, as the first query input. This is accomplished by the multiple qubit Hadamard primitive program $|P^{\mathrm{H}}_{n,1}\rangle$ defined in Eq. (6). The next step is to perform an oracle query. The following program performs an oracle call with query data prepared in $D(n:1)$:

$$|P_m\rangle \stackrel{\mathrm{def}}{=} |S_{n+1,n+2}\rangle|S_{n+1,n+2}\rangle, \quad (16)$$

where $D(n+1)$ is used as the oracle query enable qubit and $D(n+2)$ is initialized to $|1\rangle$. $D(n+1)$ is assumed to be initialized to $|0\rangle$ (i.e. the oracle query data is disabled at start-up). This program simply sets the query enable qubit for a single *UQC* cycle and then clears it, returning the state of $D(n+2:n+1)$ back to the original state. Thus, upon running $|P_m\rangle$, the result of the oracle call is in $D(n:1)$, i.e. this program is functionally equivalent to U_m.

The next step is to implement a program $|P_s\rangle$ that performs the reflection of a given state about the superposition of all basis states $|s\rangle$. This requires a conditional-phase operation that works as follows:

$$|x\rangle \rightarrow \begin{cases} |x\rangle & \textit{if } x=0, \\ -|x\rangle & \textit{otherwise} \end{cases}$$

where $|x\rangle$ is n qubits wide. Up to a global phase, this can be implemented using the following procedure:

1. Apply the σ_x operation to all n qubits.
2. Apply a controlled-**Z** operation using $n-1$ qubits as control qubits and the remaining qubit as the data qubit.
3. Apply the σ_x operation to all n qubits.

We can construct a multiple qubit controlled-**Z** program $|P^{\mathrm{CZ}}_{i,j,k}\rangle$ where qubits i through j are the control qubits and qubit k is the data qubit, with the $|P^{\mathrm{CZ}}_{i,j}\rangle$ program defined in Eq. (10) and the Toffoli program $|P^{\mathrm{Toff}}_{i,j,k}\rangle$ that we defined in [5] using a procedure analogous to that described in [6], Chapter 4. Armed with $|P^{\mathrm{CZ}}_{i,j,k}\rangle$, we construct $|P_s\rangle$ as follows:

$$|P_s\rangle \stackrel{\mathrm{def}}{=} |P^{\mathrm{H}}_{n,1}\rangle|P^{\mathrm{X}}_{n,1}\rangle|P^{\mathrm{CZ}}_{2,n,1}\rangle|P^{\mathrm{X}}_{n,1}\rangle|P^{\mathrm{H}}_{n,1}\rangle. \quad (17)$$

It can be readily verified that this is functionally equivalent to the $2|s\rangle\langle s|-\mathbb{I}$ operator. Thus, a program that performs a single Grover iteration is

$$|P_G\rangle \stackrel{\mathrm{def}}{=} |P_m\rangle|P_s\rangle. \quad (18)$$

In summary, the complete program to search a database of 2^n items with a single marked solution is

$$|G\rangle \stackrel{\mathrm{def}}{=} |P^{\mathrm{H}}_{n,1}\rangle(|P_G\rangle)^{N_G}|P_h\rangle, \quad (19)$$

where $N_G = \frac{\pi}{4}\sqrt{2^n}$ is the number of Grover iterations that can be pre-computed based on the database size, or that *UQC* can compute from the database size using a classical algorithm. Upon execution of $|G\rangle$, a measurement of $D(n:1)$ reveals the solution $|m\rangle$. Because there are no oracle queries associated with $|P^{\mathrm{H}}_{n,1}\rangle$ and $|P_h\rangle$, we immediately identify the complexity (as a measure of the number of oracle queries) of $|G\rangle$ as N_G. As is to be expected, this complexity is identical to the number of oracle queries associated with an implementation in the gate array framework.

Discussion

We have presented a scheme to allow universal quantum computers to utilize networked quantum resources. We have illustrated the scheme by devising *UQC* programs that implement the well-known oracle based Deutsch, Deutsch-Jozsa, and Grover algorithms using networked quantum oracle devices. We have therefore demonstrated that universal quantum computers can access networked quantum devices in a way analogous to that by which classical computers access network resources. The method that we used to map quantum algorithms onto *UQC* can be applied to implement and analyze other quantum algorithms.

Author Contributions

Wrote the paper: AL. Assisted in the development and analysis of the universal quantum computer: ML LvS.

References

1. Kimble HJ (2008) The quantum internet. Nature 453: 1023–1030.
2. Curcic T, Filipkowski ME, Chtchelkanova A, D'Ambrosio PA, Wolf SA, et al. (2004) Quantum networks: from quantum cryptography to quantum architecture. SIGCOMM Comput Commun Rev 34: 3–8.
3. Griffiths DJ (2005) Introduction To Quantum Mechanics. Pearson Education Inc., second edition.
4. Deutsch D (1985) The church-turing principle and the universal quantum computer. Proc R Soc London 400: 97–117.
5. Lagana AA, Lohe MA, von Smekal L (2009) Construction of a universal quantum computer. Phys Rev A 79: 052322.
6. Nielsen MA, Chuang IL (2000) Quantum Computation and Quantum Information. Cambridge University Press.
7. Waser S, Flynn MJ (1982) Introduction To Arithmetic for Digital Systems Designers. Holt, Reinhart and Winston.
8. Goldwasser S, ed (1994) Proceedings of the 35th Annual Symposium on the Foundations of Computer Science. IEEE Computer Society.
9. Grover LK (1997) Quantum mechanics helps in searching for a needle in a haystack. Phys Rev Lett 79: 325–328.