Implementing quantum algorithms using classical electrical circuits: Deutsch, Deutsch-Jozsa and Grover

Charles Peter Pentland Dyson

M.Sc. (by research)

University of York

Department of Mathematics

March 2011

Abstract

We develop the means to implement Deutsch's algorithm, the Deutsch-Jozsa algorithm and Grover's algorithm using electronic circuits, and review previous attempts to implement quantum algorithms classically. We attempt to demonstrate that these are not fundamentally quantum algorithms, but merely algorithms that scale exceptionally well on quantum computers. Finally, we discuss a prototype design for an electrical Hadamard gate and report experimental results.

Contents

Acknowledgements

The author wishes to thank Dr Stefan Weigert, Dr Chris Fewster and Prof Paul Busch for patient guidance and support.

Declaration

None of the following has been presented by the author in a previous submission.

1 Introduction

Quantum computation is the art of using quantum mechanics to perform computational tasks, a task made difficult by the eccentricities of the quantum world. Despite this, a number of examples of quantum algorithms exist, that appear to exhibit reduced computational complexity – and thus are faster than – their classical equivalents.

The prototypical example is the *Deutsch algorithm* [4]. It is perhaps the simplest algorithm demonstrating such a quality. Here, we are given one of the four possible functions $f : \mathbb{Z}_2 \to \mathbb{Z}_2$, without being told which one, and we must compute $f(0) + f(1) \mod 2$ without explicitly computing either value of $f(\cdot)$. This is made possible not by the collapse of a wavefunction (no such collapse occurs), nor the use of complex coefficients (they are not present), but by superposition alone.

Classical computation is generally performed on digital computers, in a world of sharply-defined 0s and 1s, that may permit crude parallelism (by having more than one processing unit), but lack the capacity for true superposition of states.

We propose the following: the Deutsch algorithm depends only on facilities afforded by *analogue* computation; no quantum-mechanical phenomena are required to realise it, and that there are forms of analogue computation that afford an effective alternative to the use of quantum-mechanical superposition.

We present an electrical implementation of Deutsch's algorithm, and generalise it to the Deutsch-Jozsa algorithm [5]. We then show that both algorithms may be greatly simplified when the stringencies of quantum mechanics are withdrawn, and indeed that more information may be ascertained than the quantum implementation permits. We also consider the work of Calude *et al.* [3, 1, 2] at 'dequantising' these two algorithms.

Finally, we turn our attention to Grover's algorithm [6], commonly known as the quantum search algorithm, and show that without quantum restrictions, it requires only a single iteration to deliver the required result, rather than $O(2^{\frac{n}{2}})$ iterations.

2 Digital computers

We begin, for later comparison, with a summary of the properties of digital computer.

A *bit* is an element of the set $\mathbb{Z}_2 = \{0, 1\}$. A digital computer is a device acting on a state space $S = {\mathbb{Z}_2}^{\times N}$, for a fixed natural number N – i.e. the set of all sequences of 0s and 1s of length N. A computer is provided with an *initial state* $s_{\text{in}} \in S$, which it then passes through a series of transformations, known as *gates*. The result is known as the *output* state: $s_{\text{out}} \in S$.

A gate is any map $S \to S$. There are no restrictions; in particular gates need not be reversible. For example, let us fix $N = 4$ and let a, b, c, d be elements of $\mathbb{Z}_2$. An AND gate acting on the second and third bits, could then be any one of

- $(a, b, c, d) \mapsto (a, b \text{ AND } c, X, d)$
- $(a, b, c, d) \mapsto (a, X, b \text{ AND } c, d)$
- $(a, b, c, d) \mapsto (a, b \text{ AND } c, b \text{ AND } c, d)$

for X either 0 or 1. Notice we have a choice as to where to store the result and what to put in the new, spare bit.

Another example that we will revisit frequently, is the controlled-NOT gate $(a, b) \mapsto (a, a \oplus b)$, where by $\oplus$ we denote addition modulo 2.

We assume that no information is lost by reading the output state (or indeed any intermediate state between gates), however the gates themselves may lose information, e.g. the trivial gate $S \ni s \mapsto (0, \ldots, 0)$.

Example We represent the integers 0 to $N-1$ in binary with elements in S, with the left-most bit most significant – e.g. $(1, 0, 0, \ldots, 0) \sim 2^{N-1}$. Fix N. Define the right-shift gate by $(a_N, a_{N-1}, \ldots, a_1) \mapsto (0, a_N, a_{N-1}, \ldots, a_2)$. This gate then divides its input by two, ignoring any remainder. For example, the number 23 undergoes the mapping

$$(1, 0, 1, 1, 1) \mapsto (0, 1, 0, 1, 1), \tag{1}$$

which is 11. Note that the outcome is deterministic, and that information is lost in exactly half of all cases.

3 Quantum computation

Having discussed the fundamental elements of digital computation, we recall the elements of quantum computation, drawing comparison appropriately.

3.1 Qubits and states

Just as bits are the building blocks of digital computers, so quantum bits – *qubits* – are the building blocks of quantum computers. A qubit may be in a state $|0\rangle$, $|1\rangle$, or in any complex linear combination of the two: $\alpha|0\rangle + \beta|1\rangle$. Qubits enjoy a continuum of possible states, compared to just two for ordinary bits.

The state space of an N-qubit quantum computer consists of the set of possible tensor products of N qubit states. For example, the state of a two qubit quantum computer is a complex linear combination of the states $|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle$.

As a shorthand we write $|0\rangle \otimes |1\rangle$ as $|0\rangle|1\rangle$ or even $|01\rangle$. When we write $|x\rangle$, for a natural number x, we understand $|x\rangle$ to be the tensor product of the binary expansion of x – e.g. for $x = 5$ we would have $|101\rangle$. This convention does not apply to Greek letters.

States $|x\rangle$ are called *computational basis states*; all other states are linear combinations of these and are called superposition states; these may be written as $|\psi\rangle = \sum_x \alpha_x |x\rangle$ where x is taken to be the binary representation of the integers 0 to $2^N - 1$. The state space carries an inner product structure: let $|\psi\rangle = \sum_x \alpha_x |x\rangle$ and $|\varphi\rangle = \sum_x \beta_x |x\rangle$, then $\langle\psi|\varphi\rangle = \sum_x \alpha_x^* \beta_x$.

States cannot be directly observed in general. Instead, the state $|\psi\rangle = \sum_x \alpha_x |x\rangle$ collapses to $|x\rangle$ with probability $|\alpha_x|^2 / \langle\psi|\psi\rangle$, when performing a measurement in the computational basis.

The states $|\psi\rangle$ and $\lambda|\psi\rangle$ for $\lambda \in \mathbb{C}$ and $\lambda \neq 0$ are said to be physically equivalent because the probabilities of the different outcomes are identical. We demand states be normalised – it is not physically meaningful to allow their moduli to vary. We thus stipulate that any state $|\psi\rangle$ must satisfy $\langle\psi|\psi\rangle = 1$. If a general state is written $|\psi\rangle = \sum_x \alpha_x |x\rangle$, then we require $\sum_x |\alpha_x|^2 = 1$.

3.2 States and linear algebra

We may represent the state $|\psi\rangle$ of an N-qubit quantum computer with an element $\psi \in \mathbb{C}^{2^N}$ called a *state vector*. For example the state $\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$ could be represented

as $(\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11})^{\mathrm{T}}$. In particular, the computational basis is represented as

$$|0\rangle \sim \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \tag{2a}$$

$$|1\rangle \sim \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2b}$$

We freely associate ψ with $|\psi\rangle$ from now on. The inner product in this representation is just the standard inner product on $\mathbb{C}^{2^N}$ – i.e. $(\psi, \varphi) = \psi^\dagger \varphi$.

3.3 Gates

For an N-qubit quantum computer, a gate U is a bijection on the state space that preserves inner product: $\langle U\psi | U\psi' \rangle = \langle \psi | \psi' \rangle$. They are most easily defined by reference to state vectors: in this way they are represented by endomorphisms on $\mathbb{C}^{2^N}$ – i.e. unitary matrices.

As a consequence, all gates are reversible. Digital gates such as AND, OR and the bitshift operator are thus not permitted. This is a limitation, but as we will see, it is outweighed by the capacity to turn computational states in to superposition states and back again.

The simplest example of a quantum gate is the identity gate: $|\psi\rangle \mapsto |\psi\rangle$. An example familiar from the digital world is the quantum-NOT gate T: $|0\rangle \mapsto |1\rangle$, $|1\rangle \mapsto |0\rangle$. This is also written as $|x\rangle \mapsto |\neg x\rangle$, where $\neg$ is the classical NOT operation. In state-vector notation, also known as *matrix*-notation, this is written as the matrix

$$T = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{3}$$

The controlled-NOT gate can be redefined to act on quantum states as $|ab\rangle \mapsto |a\rangle|a \oplus b\rangle$ for $a, b \in \mathbb{Z}_2$ or explicitly as:

$$|00\rangle \mapsto |00\rangle; \tag{4a}$$

$$|01\rangle \mapsto |01\rangle; \tag{4b}$$

$$|10\rangle \mapsto |11\rangle; \tag{4c}$$

$$|11\rangle \mapsto |10\rangle. \tag{4d}$$

In matrix form this is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{5}$$

Clearly for $M \leq N$ we can embed M-qubit gates into N-qubit computers by choosing which qubits to operate on. For example we can perform the controlled-NOT on a three-qubit computer by sending $|abc\rangle \mapsto |a\rangle|a \oplus b\rangle|c\rangle$. Here we would say the first qubit is the *source* or *control* and the second qubit the *target*, while the third passes through unchanged.

We may combine N- and N'-qubit gates in to an $(N + N')$-qubit gate by taking the tensor product. For example, the gate $T \otimes I$ would act as $|0\rangle|a\rangle \mapsto |1\rangle|a\rangle$, $|1\rangle|a\rangle \mapsto |0\rangle|a\rangle$ for $a \in \mathbb{Z}_2$. The matrix for such a gate is of course just the matrix tensor product, also written $T \otimes I$.

Gates may of equal dimension may be chained together. Just as in matrix multiplication and function composition, the order in which gates appear is the reverse of the order of execution. For example, consider the composition of the controlled-NOT followed by a quantum-NOT on

the second qubit, given by $\psi \mapsto T_2 C \psi$, where C is the matrix specified in (5) and T_2 is given by $I \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. Intuitively, we might expect that this is equivalent to doing a quantum-NOT on the *first* qubit, and then performing a controlled-NOT, and this conclusion is borne out by explicitly computing the matrix products.

We refer to a finite, predetermined sequence of quantum gates as a *quantum circuit*. We refer to the state to which this sequence is applied as the *initial state* and the resulting state as the *final state* or *output state*.

3.4 The Hadamard gate

This is one of the vital ingredients in quantum computation. Acting on one qubit, it sends

$$|0\rangle \mapsto |+\rangle \coloneqq (|0\rangle + |1\rangle)/\sqrt{2}, \tag{6}$$

$$|1\rangle \mapsto |-\rangle \coloneqq (|0\rangle - |1\rangle)/\sqrt{2}. \tag{7}$$

It may therefore be written out in the computational basis as the matrix

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{8}$$

We note that $H = H^\dagger$, thus since all gates are unitary we have $H^2 = I$. This gate sends computational states to superpositions and back again, paving the way for apparent parallel computation.

3.5 Oracles

An *oracle*, sometimes known as a *black box*, is a gate of known size but unknown specification, that may be subject to one or more known rules.

Example 1 Let F be the set of four functions $\mathbb{Z}_2 \to \mathbb{Z}_2$, and in particular write f_{ij} for the element of that set with $f_{ij}(0) = i$ and $f_{ij}(1) = j$.

Let U_f for $f \in F$ be a quantum gate acting on two qubits, that acts as $|x, y\rangle \mapsto |x, y \oplus f(x)\rangle$. We write U_{ij} as a shorthand for $U_{f_{ij}}$. It is clear that $U_{00} = I$ (the identity), U_{01} is the controlled-NOT, U_{10} is the controlled-NOT with the qubits swapped and U_{11} is $|x, y\rangle \mapsto |x, \neg y\rangle$, where $\neg$ is the classical NOT operation. We could also write this as $U_{11} = I \otimes T$.

It easily shown [9] that for any of the U_{ij}, we have

$$U_{ij}\left[\frac{|00\rangle + |10\rangle}{\sqrt{2}}\right] = \frac{|0\rangle|f(0)\rangle + |0\rangle|f(1)\rangle}{\sqrt{2}}. \tag{9}$$

So each oracle U can produce a state containing the complete description of the f from which it is determined. If were able to observe such a state, we would know which f had been chosen. If we were to measure it, it would collapse immediately to either $|00\rangle$ or $|01\rangle$, which would rule out some of the possible values of f, but not conclusively determine it.

4 Deutsch's algorithm

Deutsch's algorithm, and its extension – the Deutsch-Jozsa algorithm – are regarded as the prototypical examples that demonstrate that quantum information is in some way faster than classical computation [4, 5, 9]. We begin with a review of the problem Deutsch's algorithm solves:

- As before, let F be the set of four functions $\mathbb{Z}_2 \to \mathbb{Z}_2$, and denote by f_{ij} the element of that set with $f_{ij}(0) = i$ and $f_{ij}(1) = j$.
- Alice selects an $f \in F$, and gives Bob an oracle that that implements the chosen f, without telling Bob which f was selected. For example, this could be a calculator-like device with two buttons: 'show $f(0)$' and 'show $f(1)$'.
- Bob must compute $f(0) \oplus f(1)$ where, as before, we use $\oplus$ to mean addition modulo 2. We say f is *balanced* if $f(0) \neq f(1)$ and constant otherwise.

A classical digital solution is obvious: Bob uses the oracle to extract $f(0)$, then uses it again to extract $f(1)$, then determines $f(0) \oplus f(1)$. We say 'determines' rather than 'computes', because it is quite reasonable to assume that the four possible outcomes of $f(0) \oplus f(1)$ are known to Bob in advance. This requires two invocations of the oracle in all.

The quantum-mechanical solution – Deutsch's algorithm – requires only a single invocation of the oracle:

- Alice selects an $f \in F$ as before, and gives Bob the corresponding quantum oracle U_f as defined in Example 1.
- Bob constructs a quantum computer by chaining together U_f with two other gates in the following order
$$H \otimes H \to U_f \to H \otimes I, \tag{10}$$
and sets its initial state $|\psi_{\text{in}}\rangle$ to $|01\rangle$.
- The output may then be shown to be one of the two physically-equivalent states
$$|\psi_{\text{out}}\rangle = \pm\frac{1}{\sqrt{2}}|f(0) \oplus f(1)\rangle(|0\rangle - |1\rangle). \tag{11}$$
- A measurement is then performed on the first qubit, which collapses to $|f(0) \oplus f(1)\rangle$, thus delivering the sought result.

Qualitatively: both input qubits $|0\rangle$ and $|1\rangle$ are symmetrised with Hadamard gates, sent through U_f, then the first qubit is sent through another Hadamard gate. From the measurement, we obtain a global property of f, which in the classical digital case would require two separate evaluations of the function. Alternatively: Bob determines which U_f Alice used with a single invocation, rather than two: *both f(0) and f(1) appear to have been evaluated simultaneously during the execution of the algorithm.* It is in this sense that Deutsch's algorithm is said to be faster than its classical equivalent.

Unlike some other quantum algorithms, we notice that the full capabilities of the quantum-mechanical arsenal do not appear to be employed. In particular, at no stage were complex numbers used. Further, the measurement of the first qubit (the only one to carry useful information) is entirely deterministic: it has already attained its final state when measurement is performed.

What lends the algorithm its ability to know simultaneously both values of $f(\cdot)$ (even though it can only produce a global property of them, not the individual values) is the power of superposition.

Since superposition is not unique to quantum mechanics, we seek other systems within which Deutsch's algorithm may be implemented.

5 Real quantum mechanics

We simplify the task of finding alternative implementations of Deutsch's algorithm by dispensing with complex numbers. We note that Deutsch's algorithm is not unique in not natively requiring complex numbers – Grover's algorithm in particular makes no use of them.[1]

Henceforth, if a qubit is in a state that may be expressed as a *real* linear combination of $|0\rangle$ and $|1\rangle$, we call it a *rebit*. A state with real coefficients of all the $|x\rangle$ we call a *real state* and we will write e.g. $|\psi]$ to distinguish such states. Thus in particular we might decompose a state as $|\psi] = \sum_x \alpha_x |x]$, where each $\alpha_x \in \mathbb{R}$. The coefficients might instead be written $\alpha_x = [x|\psi]$ as we effectively inherit a real inner product.

We also inherit all the machinery of quantum gates, which retain their definition but now act on a real state space. In the state vector representation, they are real orthogonal matrices, rather than Hermitian as before.

6 An electrical implementation of the Deutsch algorithm

Since Deutsch's algorithm requires no complex numbers, it can be implemented using two rebits. At each stage of the calculation – initial, in between each pair of gates and final – the state may be written

$$|\Psi] = [00|\Psi]\,|00] + [01|\Psi]\,|01] + [10|\Psi]\,|10] + [11|\Psi]\,|11] \tag{12}$$

If we were to simulate or mentally compute the algorithm, we would need to keep track of four real numbers $[x|\psi]$ at each stage in the calculation, for each x.

It is possible to represent these four real numbers with voltages, and to model their changes by passing current through electric circuits. We refer to each coefficient as a *wire* or *line* in an electrical context, and make the following physical assumptions:

1. All circuits used share a common electromotive supply $V_{\mathrm{sup}\pm}$. This supply exceeds the maximum voltage that might be needed to represent a coefficient at any point in the system.

2. Voltages may be measured without influencing the system significantly.

3. There exist analogue *summing gates* Σ, that take two or more wires as inputs and output one wire, with voltage set to the algebraic sum of the input voltages. Their operation is entirely linear and they draw a negligible amount of current.

4. There exist analogue *multiplicative gates* that take one wire of voltage V and output a voltage λV for fixed $\lambda \in [-1, 1]$. In particular we call the gate with $\lambda = -1$ the *analogue-NOT gate*. If $0 < \lambda < 1$ then we call the gate an *attenuating gate*.

5. Both summing and multiplicative gates are able to source a reasonable amount of current without inducing any detectable voltage drop.

Note that a multiplicative gate is not a quantum-mechanical gate: it acts on a single wire (not a state), and cannot be represented by a unitary matrix. In particular, the analogue-NOT gate is to be distinguished from the quantum-NOT gate.

All of these assumptions are physically realistic, and implementation details are given in Appendix A.

[1] It is actually very easy to remove complex numbers from algorithms that really do employ them, by simply adding an ancillary qubit effectively take the place of i – see [11].

By way of notation, we represent multiplicative gates by a rounded box encircling the multiplication factor:

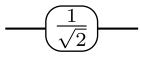

Figure 1

Summing gates are represented by an encircled Σ:

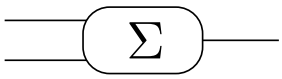

Figure 2

As a special case, we represent gates that multiply by a factor of -1 by a small circle on another gate's input. This is analogous with the convention in digital circuitry for representing digital negation:

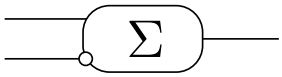

Figure 3

6.1 The Hadamard gate

In this context, a real Hadamard acting on a wavefunction $H : |\psi] \mapsto |\psi']$ is merely a map between voltages representing the two numbers $[0|\psi]$ and $[1|\psi]$ – the map in question is:

$$[0|\psi] \mapsto \left([0|\psi] + [1|\psi]\right)/\sqrt{2} \;=\; [0|\psi']\,, \tag{13a}$$

$$[1|\psi] \mapsto \left([0|\psi] - [1|\psi]\right)/\sqrt{2} \;=\; [1|\psi']\,. \tag{13b}$$

This is readily implemented as a circuit:

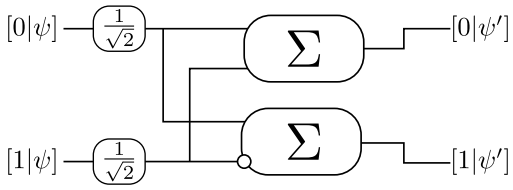

Figure 4

The circuit is to be read left-to right. $[0|\psi]$ and $[1|\psi]$ are the two inputs while $[0|\psi']$ and $[0|\psi']$ are the outputs. It is worth recalling the convention that wires meeting at a cross intersection do *not* represent an electrical connection between the two.

The following is then clear:

- Setting the input $[0|\psi] = 1$, $[1|\psi] = 0$ yields the expected result $[0|\psi'] = [1|\psi'] = 2^{-\frac{1}{2}}$.
- Setting the input $[0|\psi] = 0$, $[1|\psi] = 1$ yields the expected result $[0|\psi'] = -[1|\psi'] = 2^{-\frac{1}{2}}$.
- The action of the gate is linear (subject to reasonable electric assumptions) in the two variables.
- Concatenating two copies of the gate yields an electrical identity map.

A prototype design is detailed in Appendix B, with experimental results detailed in Appendix C. The prototype performed exactly as described, and with remarkable accuracy. We therefore explore the possibilities afforded to us by analogue Hadamard gates.

6.2 Implementing the algorithm

Being able to implement a Hadamard gate electrically is not sufficient to implement the Deutsch algorithm. We require gates that act on a pair of rebits, thus we require four wires instead of two, one for each of the real coefficients of $[00|\psi]$ to $[11|\psi]$.

To implement Deutsch's algorithm, we require the following gates:

- $H \otimes I$
- $H \otimes H$
- U_f for each $f \in F$

It is very easy to re-use our circuit for the one-qubit Hadamard gate to implement $H \otimes I$ as follows. We use an encircled H as shorthand for the entire circuit described in Figure 4:

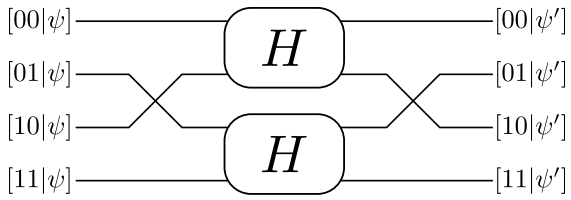

Figure 5

As a matrix, the gate $H \otimes H$ takes the form

$$\frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}. \tag{14}$$

To implement it electrically, we use four gates to attenuate the input by half, six analogue NOT gates and four Σ gates – 14 gates in all. We simply attenuate the input, then take sums with NOT gates in the appropriate places.

Alternatively, we note that $H \otimes H = (H \otimes I)(I \otimes H)$. We can implement $I \otimes H$ with the following circuit:

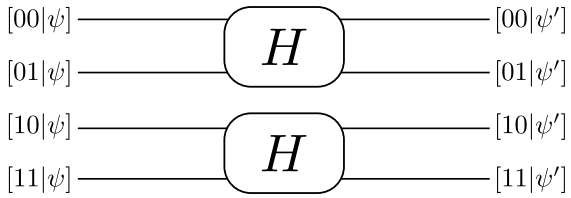

Figure 6

So the complete $H \otimes H$ circuit could be constructed thus:

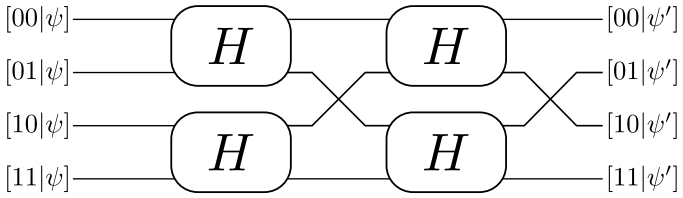

Figure 7

This implementation requires 4 lots of two attenuation gates, two summing gates and a NOT gate – 20 components in all.

Finally, we must implement all four of the U_f gates. These are in fact completely trivial as they are just permutations of $\mathbb{Z}_4$: to implement them electrically one need only perform a relabelling of the wires. For instance, to implement U_{01} (the controlled-NOT), we simply swap the labels of $|10]$ and $|11]$ while leaving the others unchanged.

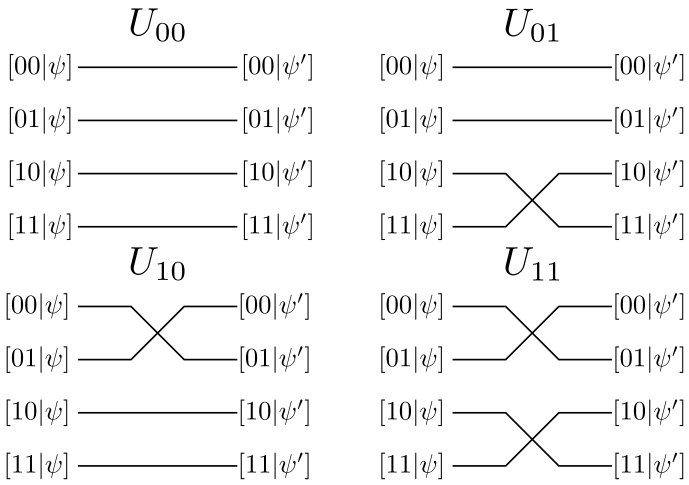

Figure 8

In fact, viewing the action of the U_{ij} in this light, we see that Deutsch's problem is actually to find whether the number of transpositions is even ($f(0) \oplus f(1) = 0$) or odd ($f(0) \oplus f(1) = 1$). We shall return to this point in the case of the Deutsch-Jozsa algorithm later.

6.3 Deutsch's algorithm – electrical implementation

Alice chooses a function $f \in F$, and gives Bob a circuit implementing U_f, without specifying which f she chose.

Bob then

- connects the following three circuits in order:

$$\mathsf{INPUT} \to H \otimes H \to U_f \to H \otimes I \to \mathsf{OUTPUT}, \tag{15}$$

- connects a voltage[2] V to the $|01]$ input of the first circuit, while holding the other three inputs at ground and
- measures the voltage at the $|00]$ output of the final circuit.

If the measured voltage is $\pm V/\sqrt{2}$, then $f(0) \oplus f(1)$ is 1; if the measured voltage is zero then that quantity is zero.

6.4 Determining f

We mentioned before that the different U_{ij} may be thought of in terms of transpositions. We expound upon that idea. Let t be 1 if $|00]$ and $|01]$ are swapped, and let t' be 1 if $|10]$ and $|11]$ are swapped, in both cases let them be zero otherwise. Thus U_{00} corresponds to $t = t' = 0$ etc.

The state of the wavefunction after the first pair of Hadamards $H \otimes H$ is

$$|\Psi] = \frac{1}{2}\left(|00] - |01] + |10] - |11]\right). \tag{16}$$

After passing through U_{ij} this becomes

$$|\varphi] = \frac{1}{2}(-1)^t(|00] - |01] + |10] - |11]) + \tag{17}$$

$$\frac{1}{2}(-1)^{t'}(|00] - |01] - |10] + |11]) \tag{18}$$

We then have the following cases:

$$2\,|\varphi] = \begin{cases} |00] - |01] & t = 0;\ t' = 0 \\ |10] - |11] & t = 0;\ t' = 1 \\ -|10] + |11] & t = 1;\ t' = 0 \\ -|00] - |01] & t = 1;\ t' = 1 \end{cases} \tag{19}$$

Thus we may expand our previous conclusion. Suppose Bob measures both $|00]$ and $|10]$, then:

- If Bob finds $V/2$ at $|00]$ then $f(0) = 0$ and $f(1) = 0$.

[2]For technical reasons the chosen voltage should lie well within $V_{\mathrm{sup}\pm}$; see Appendix A for details.

- If Bob finds $-V/2$ at $|00]$ then $f(0) = 1$ and $f(1) = 1$.
- If Bob finds $V/2$ at $|10]$ then $f(0) = 0$ and $f(1) = 1$.
- If Bob finds $-V/2$ at $|10]$ then $f(0) = 1$ and $f(1) = 0$.

So we discover that the electronic implementation actually gives us more information than we asked for! We note that it may appear that Bob is making four separate measurements, and that it might be concluded that this invalidates the notion that the this method is faster, or more powerful, than the quantum Deutsch algorithm. However, it would be trivial for Bob to build a circuit that illuminates a single light for each of the four possibilities – in this case familiar digital logic circuitry will suffice to complete the task.

It is worth noting that all this turns the Deutsch algorithm from a modular arithmetic routine into a simple (and deterministic) search algorithm. We shall see later that a classical implementation creates new possibilities for Grover's quantum search algorithm.

6.5 Discussion

Just as with the quantum mechanical implementation of Deutsch's algorithm, Bob is able to deduce $f(0) \oplus f(1)$ with a single invocation of the oracle, something that would be impossible for a *digital* logic computer. As with the quantum implementation, the result is sharp and deterministic.

One might be tempted to dismiss this as a crude simulation of a quantum computer, but this is in fact a legitimate implementation of the algorithm in its own right. We note that existing quantum implementations of Deutsch's algorithm [9, 7] are considerably more involved.

Furthermore, because it is not constrained by quantum-mechanical limitations, we may simplify it in ways not permitted in the quantum case. As we have seen, Bob need only measure one voltage – the other three do not contribute to his findings. This means we need not implement the full $H \otimes I$ gate at the end, but only enough components to produce the $|00]$ output. Furthermore, rather than implement the $H \otimes H$ gate at the beginning, we may omit it entirely and apply an input $+V, -V, +V, -V$ to the U_f directly – we do not need the factor of one half, and producing these voltages requires no additional circuitry.

Alternatively, if Bob is prepared to measure two voltages, he is even able to determine f exactly – something beyond the reach of the original algorithm.

Thus, not only can we implement Deutsch's algorithm without a quantum computer, we can do so either with considerable simplification, or with greater results.

6.6 Existing work

In [3] Calude *dequantises* (his term) the Deutsch algorithm by considering the same problem expressed on the space $\mathbb{Q}[\mathrm{i}] = \{a + b\mathrm{i} : \ a, b \in \mathbb{Q}\}$. The functions f_{ij} are now thought of as functions on this space, rather than on $\{0, 1\}$. The oracles $U_{f_{ij}}$ are replaced with functions $C_{f_{ij}}$ defined as:

$$C_{f_{ij}} : \ (a + b\mathrm{i}) \mapsto (-1)^{0 \oplus f(0)} a + (-1)^{1 \oplus f(1)} b\mathrm{i}. \tag{20}$$

We use C_{ij} as a shorthand for these functions, and it may be shown that:

$$C_{00} : \ x \ \mapsto \ \ x^*; \tag{21a}$$
$$C_{01} : \ x \ \mapsto \ \ x; \tag{21b}$$
$$C_{10} : \ x \ \mapsto -x; \tag{21c}$$
$$C_{11} : \ x \ \mapsto -x^*. \tag{21d}$$

It is then shown that $f(0) \oplus f(1) = 1$ *if and only if* $(\mathrm{i}-1)C_f(1+\mathrm{i})$ *is real*. Thus one need only evaluate a single expression to know a global property of f.

The construction arises by replacing the usual Hilbert space with a subset of $\mathbb{C}$. Since we only use the real part of $\mathbb{C}^2$ in the quantum/analogue constructions, the spaces are effectively the same size. It is immediate that an equivalent construction may be performed on $\mathbb{R}^2$ owing to the isomorphism between that space and $\mathbb{C}$. In such a construction, C_f would take the form

$$C_f = \begin{bmatrix} (-1)^{f(0)} & 0 \\ 0 & (-1)^{1+f(1)} \end{bmatrix}, \tag{22}$$

and $(\mathrm{i}-1)C_f(1+\mathrm{i})$ becomes

$$\begin{bmatrix} (-1)^{f(0)+1} + (-1)^{f(1)+1} & (-1)^{f(0)} + (-1)^{f(1)+1} \\ (-1)^{f(0)+1} + (-1)^{f(1)} & (-1)^{f(0)+1} \end{bmatrix}. \tag{23}$$

The condition $(\mathrm{i}-1)C_f(1+\mathrm{i})$ is real is thus equivalent to (recalling that the off-diagonal elements play the role of the imaginary parts) $(-1)^{f(0)+1} + (-1)^{f(1)} = 0$, which in turn is the same as $f(0) \oplus f(1) = 0$.

We note for comparison that the explicit formula for U_f is

$$\begin{bmatrix} 1-f(0) & f(0) & 0 & 0 \\ f(0) & 1-f(0) & 0 & 0 \\ 0 & 0 & 1-f(1) & f(1) \\ 0 & 0 & f(1) & 1-f(1) \end{bmatrix}, \tag{24}$$

thus this is not quite a literal translation of the original idea (as this matrix is not the same as $I \otimes C_f$ or something similar), though the principle of operation is much the same.

Calude's notion of *dequantisation* implies that the algorithm may be deployed classically in some sense. However, translating this algorithm back in to the space $\mathbb{R}^2$, it becomes:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \to \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \to C_f \to \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \to \mathsf{OUTPUT}, \tag{25}$$

where the output vector is:

$$\begin{pmatrix} (-1)^{f(0)+1} + (-1)^{f(1)+1} \\ (-1)^{f(0)} + (-1)^{f(1)+1} \end{pmatrix}, \tag{26}$$

which appears to be very similar to the original algorithm. Just as with the quantum Deutsch algorithm, one starts with a vector, transforms it, sends it through an oracle, then transforms it back to something more tangible. The resulting vector takes distinct values for each of the four possible f_{ij}, and thus the algorithm actually determines f completely.

So to actually execute Calude's algorithm requires matrix multiplication (or equivalently complex arithmetic), in much the same way as computational simulation of the quantum Deutsch algorithm. Thus it is hard to see how it is any more dequantised than a matrix interpretation of the Deutsch algorithm – it is merely a refinement.

Finally, we note that Calude asserts that Deutsch's algorithm is non-deterministic. This is not strictly accurate: while a quantum (or indeed electrical) implementation of the algorithm will yield a small amount of error in the final result, the algorithm itself produces entirely sharp and deterministic output.

During preparation, we became aware of the work of the pre-existing work of Kaan [8], noting that Hadamards and hence the Deutsch and Deutsch-Jozsa algorithms could be implemented electrically. The document appears to be incomplete however, as a full implementation of a Hadamard gate is not shown.

7 The Deutsch-Jozsa algorithm and scaling

We note that solving Deutsch's algorithm on a computer is just evaluating the matrix product $ABCv$. We might be tempted to work out the product ABC and then apply v, resulting in many unnecessary calculations.

In fact, we only need to know the first row of the result, so we evaluate $A_{1b}B_{bc}C_{cd}v_d$. This is the sum of $4^3 = 64$ products of four numbers, 192 operations in all; hardly an efficient classical algorithm it would seem.

The worth of an algorithm really depends not on how many operations it requires in one scenario, but how it scales with the amount of data to be processed. To this end, we recall the Deutsch-Jozsa algorithm, which is a generalisation of the Deutsch algorithm.

This time we have $n + 1$ qubits, rather than the original two. The function $f : \mathbb{Z}_{2^n} \to \mathbb{Z}_2$ is either balanced (zero for exactly half its input) or constant (always zero or always one).

The oracle U_f keeps its definition as the map $|x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$, but this time $|x\rangle$ is an n-qubit register – i.e. $0 \leq x < 2^n$. Now if we fix a particular $x = x_0$, this is either the map

$$|x_0\rangle|0\rangle \mapsto |x_0\rangle|0\rangle,\ |x_0\rangle|1\rangle \mapsto |x_0\rangle|1\rangle, \tag{27a}$$

if $f(x_0) = 0$, or the quantum-NOT if $f(x_0) = 1$:

$$|x_0\rangle|0\rangle \mapsto |x_0\rangle|1\rangle, \quad |x_0\rangle|1\rangle \mapsto |x_0\rangle|0\rangle. \tag{27b}$$

So, as before, U_f could be described in terms of permutations. Because of the restrictions on f either:

- U_f is the identity;
- $U_f|x\rangle|y\rangle = |x\rangle|\neg y\rangle$;
- $U_f|x\rangle|\cdot\rangle$ is the quantum-NOT for exactly half the possible values of x.

The first two occur when f is constant, the final possibility when f is balanced. If we wish to determine which is the case digitally, the obvious solution is to work out if $U_f|0\rangle|\cdot\rangle$ is the identity or a quantum-NOT, then $U_f|1\rangle|\cdot\rangle$ etc. until one of the following conditions occurs:

- After performing the experiment from $x = 0$ to $x = 2^{n-1} + 1$, we only see one type of map, and conclude f is constant;
- We see both types of map in the first 2^{n-1} examinations, and conclude f is balanced.

This requires at least two and at most $2^{n-1} + 1$ invocations of the oracle.

7.1 The Deutsch-Jozsa algorithm

The quantum solution to the problem is a simple extension of the quantum Deutsch algorithm. This time we have $n + 1$ qubits. f and U_f are as above. The initial state is $|0\rangle^{\otimes n}|1\rangle$, which is then passed through $H^{\otimes n+1}$, U_f and $H^{\otimes n} \otimes I$ successively:

$$|0\rangle^{\otimes n}|1\rangle \to H^{\otimes n+1} \to U_f \to H^{\otimes n} \otimes I \tag{28}$$

We ignore the final qubit, commonly called the *ancilla* (which as before is in the state $(|0\rangle - |1\rangle)/\sqrt{2}$), and concentrate on the initial n, which at the end of the procedure will be in the state [9]:

$$2^{-n} \sum_{x,y} (-1)^{x \cdot y + f(x)} |y\rangle \tag{29}$$

Here, $x \cdot y$ means the bitwise product of the two numbers – e.g. if $x = 6 = 110\text{b}$ and $y = 3 = 2 + 1 = 011\text{b}$, then $x \cdot y = 0 + 1 + 0 = 1$.

We write $|\mathbf{0}\rangle$ for the $|00\ldots0\rangle$ computational basis state of the n-qubit register. Examining the coefficient of $|\mathbf{0}\rangle$:

$$2^{-n}\sum_x (-1)^{f(x)}, \tag{30}$$

we see that if f is balanced, all the ±1 contributions cancel, and thus the coefficient is zero, so the final state is *not* $|\mathbf{0}\rangle$. Thus we conclude: after measurement, f is balanced if and only if the final state is $|\mathbf{0}\rangle$ up to a phase factor.

We note that as before no use of complex numbers is made.

7.2 The electrical implementation

The Deutsch-Jozsa algorithm is of course readily translated into an electrical algorithm. We note that

$$H^{\otimes n} = \underbrace{(H \otimes I \otimes I \otimes \cdots \otimes I)(I \otimes H \otimes I \otimes \cdots \otimes I)\ldots}_{n \text{ factors}} \tag{31}$$

As is evident from previous diagrams, each of factors terms individually requires n Hadamard gates to implement electrically. We would thus require n^2 one-rebit Hadamards just to implement $H^{\otimes n}$!

With that in mind, we take advantage of the many conveniences of the electrical world: we note that since $H^{\otimes(n+1)}|\mathbf{0}\rangle|1\rangle = (H|0\rangle)^{\otimes n}H|1\rangle = |+\rangle^{\otimes n}|-\rangle$, we may skip the initial $H^{\otimes(n+1)}$ and simply apply $2^{-n/2}(+1,-1,+1,-1,\ldots)$ as our input state. Electrically this is just the sequence of voltages $+V,-V,+V,-V,\ldots$, which we apply directly to U_f. As before, the attenuating factor is not important. This allows us to perform the calculation using just the $2n^2$ one-rebit Hadamards needed to make $H^{\otimes n} \otimes I$.

7.3 Existing work

In [2], Abbott and Calude consider the three-qubit case of the Deutsch-Jozsa algorithm, extending the presentation in [3]. In particular, C_f is now a map $\mathbb{C}^2 \to \mathbb{C}^2$:

$$C_f : \begin{pmatrix} a_1 + b_1\mathrm{i} \\ a_2 + b_2\mathrm{i} \end{pmatrix} \mapsto (-1)^{f(00)} \begin{pmatrix} a_1 + (-1)^{f(00)\oplus f(10)} b_1\mathrm{i} \\ a_2 + (-1)^{f(10)\oplus f(11)} b_2\mathrm{i} \end{pmatrix}. \tag{32}$$

It is then noted that the value of

$$\frac{1+\mathrm{i}}{2} C_f \begin{pmatrix} 1+\mathrm{i} \\ 1+\mathrm{i} \end{pmatrix} \tag{33}$$

is sufficient to determine whether f is constant or balanced. Again, to actually implement this variation on the algorithm requires involved matrix calculations. In [1] this method is revisited, although not fully generalised to $n \geq 3$.

7.4 Scaling behaviour

As before, we are effectively computing a product of matrices, but this time the matrices are of size 2^{n+1}. Assuming we require the whole state vector at the end, we have to compute $A_{i\alpha}B_{\alpha\beta}C_{\beta\gamma}v_\gamma$, then check the values of up to half of the output vector's entries (we need only check either the even or odd entries as the final rebit has known value). So we require 2^n rows, and each row requires the computation of 2^{3n} products – an exponential scaling.

Consider the case $n = 10$. We would require three matrices of size 2048×2048 and an initial vector of length 2048. The number of products we'd need to compute is 1073741824. This is approximately 10^9 operations. Assuming each one takes 10 cycles on a 1GHz processor, that would take around 10 seconds.

A quantum computer needs $n+1$ qubits and $2n+1$ times $n+1$-qubit Hadamards; 11 qubits and 21 Hadamards in this case.

The electrical method requires (at most) 2^{n+1} lines or $n+1$ rebits – i.e. 2048 separate wires which then pass through $n^2 = 100$ one-qubit Hadamard gates. They can then be sent through a classical OR gate (something forbidden in real quantum information) and a light may be made to turn on for each of the two possible outcomes.

So both digital and analogue are exponential in some sense, the former takes 2^{3n} operations, the latter requires 2^{n+1} wires.

Consider the case $n = 20$. By the same reasoning, the digital computer would take a few hundred years to do this calculation. To do it the analogue way, we would need 2097152 lines and 400 Hadamards, pushing the limits of available silicon technology, but not perhaps completely impossible.

7.5 The Deutsch-Jozsa algorithm without Hadamard gates

Consider the state before U_f is applied. Ignoring normalisation, it takes the form:

$$\sum_{x=0}^{2^n-1} \left(|x\rangle |0\rangle - |x\rangle |1\rangle \right). \tag{34}$$

We apply U_f – whenever $f(x_0) = 1$, the sign for that term will flip. This means the state after U_f takes the form:

$$\sum_{x=0}^{2^n-1} (-1)^{f(x)} \left(|x\rangle |0\rangle - |x\rangle |1\rangle \right). \tag{35}$$

This is enough information to give us *the full specification of* f, without the use of any Hadamard gates whatsoever (we recall that they are superfluous). It does not however solve the original problem – to determine whether or not f is balanced. We will go one step further.

Suppose $f(x) = 0$ for all x. Then we would expect the electrical output to be $V, -V, V, -V, \ldots$ for some V – call this the *voltage list*. For each x such that $f(x) = 1$, the x^{th} pair of voltages is swapped from $(V, -V)$ to $(-V, V)$.

Define V_a to be the algebraic sum of the first, third, fifth, etc. elements of the voltage list. We note (see Appendix A) that such a value is easily computed electrically. We then have: $f(x) = 0$ *for* V_a/V *values of* x. Thus f is balanced if and only if $V_a/V = 2^{n-1}$.

This yields more information than the quantum solution, and requires no explicit use of Hadamard gates. We are able to determine more information about f than before, and without making use of Hadamard gates. Indeed, since U_f consists only of a switchboard of transpositions, only the voltage-summing at the end requires non-trivial circuitry.

7.6 The Deutsch algorithm without Hadamard gates

The method described above of course applies in the $n = 1$ case, thus we are able to compute the solution to Deutsch's algorithm without recourse to Hadamard gates. Recall as before that we are able to think of U_f as one of four possible transposition mappings. When implemented as an electric circuit, the U_f merely act as a relabelling of the wires. From the above discussion we have a new procedure for solving Deutsch's problem:

- Alice selects an f, and gives Bob an appropriate U_f circuit – a black box with four wires leaving from one side and four leaving from the other. The inside of the box contains one of the four valid permutations of the wires.
- Bob applies a voltage V to the first and third wires, and a voltage $-V$ to the remaining two. If the output is $0V$, then $f(0) \otimes f(1) = 1$.

How the algorithm works is immediate from the four circuit diagrams:

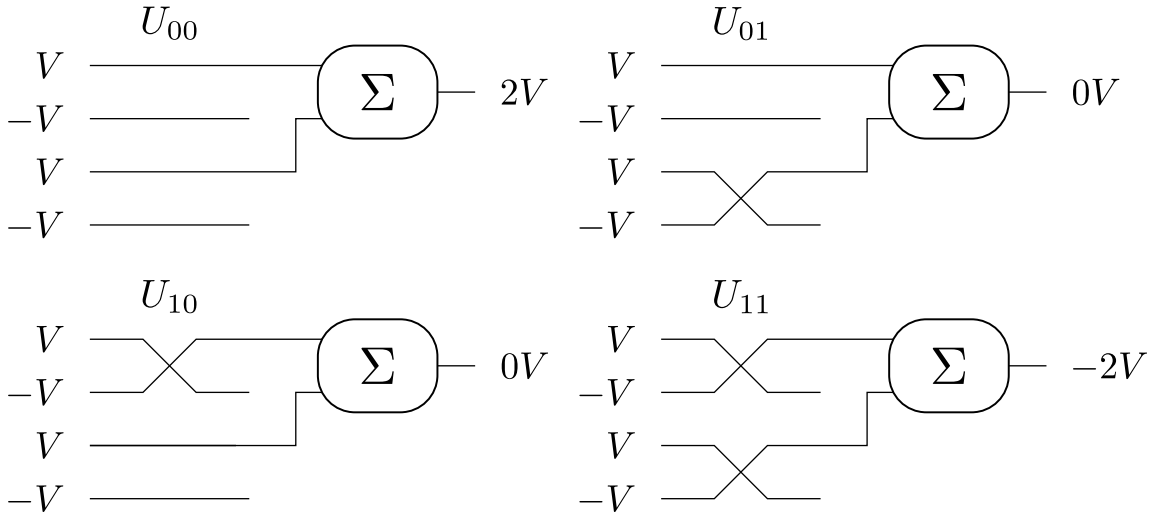

Figure 9

For the Deutsch-Jozsa problem, this algorithm of course scales so that there are 2^n input and output wires from the oracle box. Bob still applies an alternating sequence $V, -V, V, -V, \ldots$ to the inputs and if the output is $0V$ then the function is balanced.

This solution can be translated back in to the language of matrices: we note that any U_f may be expressed as

$$U_f = \bigoplus_{i=0}^{n} M_i \tag{36}$$

where each of the M_i are either the NOT-gate T (where there is a transposition) or the identity gate I (where there is none). It is evident that for any such matrix, computing

$$(1, 0, 1, 0 \ldots)^{\mathrm{T}} U_f (1, -1, 1, -1, \ldots) \tag{37}$$

yields 0 if and only if exactly half of the M_i are equal to I.

Thus for instance in the Deutsch case ($n = 2$) with f_{01} we have $U_f = I \oplus T$ and we compute

$$\begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} = 0. \tag{38}$$

As far as scaling is concerned, 2^n wires are required for a function of n values, but only one Σ box is ever required. However, summing 2^n voltages creates practical problems – the voltages

would have to be so small that electrical noise becomes a problem. Instead, voltages would have to be summed in batches, then attenuated and summed again in $O(\log 2^n) = O(n)$ stages, which means that although computation time is theoretically instant, it still requires a large number of components and a very large number of wires for large n.

8 Grover's algorithm

The quantum search algorithm, or Grover algorithm [6], is another example of a quantum algorithm with no dependence on complex numbers, and limited dependency on the notion of state collapse.

For a detailed description of the algorithm see [9]. We assume the reader is familiar with the workings of the procedure, so we merely summarise the formulation.

Let $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$ be an unknown function such that $f(y) = 1$ for some y and all other values are zero. We define the oracle gate O on $n+1$ qubits by

$$O|x\rangle|q\rangle = |x\rangle|q \oplus f(x)\rangle, \tag{39}$$

where $|x\rangle$ is an n-qubit address register, and $|q\rangle$ is a single qubit called the target register. We note that O has the following useful property:

$$O|x\rangle|-\rangle = (-1)^{f(x)}|x\rangle|-\rangle, \tag{40}$$

where as before $|\pm\rangle = 2^{-\frac{1}{2}}(|0\rangle \pm |1\rangle)$. We also define the shorthand $|\psi_n\rangle = |+\rangle^{\otimes n}$.

We define the Grover gate, G by

$$G = (2|\psi_n\rangle\langle\psi_n| - I)O. \tag{41}$$

The algorithm is then as follows:

Fix n. Alice selects an integer $0 \leq y < 2^n$ and gives Bob the oracle O corresponding to the function f with $f(y) = 1$. Bob uses an $n+1$-qubit quantum computer and

- initialises its state to $|\psi_n\rangle|-\rangle$,
- applies the Grover gate R times where $R = \left\lceil \pi\sqrt{2^n}/4 \right\rceil$,
- measures the state, taking the value of the first n qubits to be the bitstring for y.

The state before measurement is approximately $|y\rangle|-\rangle$, with high probability, though there is a non-zero probability in general the measurement will give the wrong answer. This is not always the case, however.

Example Fix $n = 2$. Then the initial state is $|+\rangle|+\rangle|-\rangle$ or just $|{+}{+}\rangle|-\rangle$. After applying the oracle O, the state is

$$\{|{+}{+}\rangle - |y\rangle\}\,|-\rangle$$

and applying the rest of G leaves us with

$$\begin{aligned}&\{2|{+}{+}\rangle\langle{+}{+}| - I\}\,\{|{+}{+}\rangle - |y\rangle\}\,|-\rangle \\ &= \{2|{+}{+}\rangle\langle{+}{+}|{+}{+}\rangle - |{+}{+}\rangle - 2|{+}{+}\rangle\langle{+}{+}|y\rangle + |y\rangle\}\,|-\rangle.\end{aligned}$$

Noting that $|{+}{+}\rangle = \frac{1}{2}(|00\rangle + \dots)$ we see that $\langle{+}{+}|y\rangle = \frac{1}{2}$ so this simplifies to $|y\rangle|-\rangle$. Thus in the $n = 2$ case the algorithm is deterministic.

Consider the general case. The initial state is

$$|\psi_n\rangle|-\rangle = \left[\alpha \sum_{x=0}^{2^n-1} |n\rangle\right] |-\rangle,$$

where we set $\alpha = 2^{n/2}$ for convenience. After we apply O the state is

$$[|\psi_n\rangle - 2\alpha|y\rangle]\,|-\rangle,$$

and after applying the rest of G we have

$$\begin{aligned}&(2|\psi_n\rangle\langle\psi_n| - I)\,(|\psi_n\rangle - 2\alpha|y\rangle)\\ &= 2|\psi_n\rangle - 4\alpha|\psi_n\rangle \underbrace{\langle\psi_n|y\rangle}_{\alpha} - |\psi_n\rangle + 2\alpha|y\rangle.\end{aligned}$$

The final state after a single iteration is therefore

$$|\varphi_{\text{out}}\rangle = [(1-4\alpha^2)|\psi_n\rangle + 2\alpha|y\rangle]|-\rangle. \tag{42}$$

The coefficients of φ_{out} are then given by

$$\langle\varphi_{\text{out}}|x,-\rangle = \begin{cases}(1-4\alpha^2)\alpha & x \neq y\\ (1-4\alpha^2)\alpha + 2\alpha & x = y.\end{cases} \tag{43}$$

One may show that the ratio of the latter quantity to the former tends to 3 as $n \to \infty$, and that the latter is always larger than the former.

Thus, *if the final state can be accurately known, the algorithm is deterministic after a single iteration.* Thus in principle an electrical implementation can complete the search in a *single* iteration.

8.1 The electrical implementation

We note that at no stage did complex numbers have a part to play in the preceding calculation, and we have shown that collapse plays no part in the calculation itself, it is merely an obstacle to be overcome. From our previous consideration of Deutsch's algorithm, it is clear that the algorithm may be implemented with an electric circuit – we only require a single Grover gate, and this may be implemented easily enough with the techniques already described.

The initial state is $|\psi_n]\,|-]$. Ignoring normalisation, this is $(1,-1,1,-1,\dots)$. The output is a sequence of pairs $(V,-V)$, except for the y^{th} pair, which will be greater in magnitude. The algorithm is then specified as follows:

Let n be fixed for all time. Alice chooses a suitable function f, and gives Bob an electrical implementation of the respective oracle O. Bob builds an n-rebit computer by connecting the following circuits:

$$\mathsf{INPUT} \to O \to (2\,|\psi_n]\,[\psi_n\,| - I) \to \mathsf{OUTPUT}, \tag{44}$$

where the input is the set of numbers $(1,-1,1,-1,\dots)$. The output will all be $\pm V$ for some fixed number V, except for the $2y-1$ and $2y^{\text{th}}$ elements, which will be substantially (~ 3 times for large n) greater in magnitude.

We note that Bob need not take out his multimeter and test each and every voltage (which would somewhat undermine the idea of using a search algorithm!) Instead, simple circuitry could be inserted that lights an appropriate lamp only when the voltage is sufficiently high. An array of LEDs have the convenient property of only illuminating above a threshold voltage, and thus would be ideal. This is purely a matter of display, not of searching.

8.2 Discussion

In both the quantum and electrical case, the value to be found is already known, and contained within, the oracle. This is quite different to looking for the solution to an equation – indeed, it bears better comparison with picking a lock.

Whoever builds the oracle – quantum or electrical – must already know the value of y. It is therefore not entirely surprising that it can be deduced in a single iteration. In fact, in the electrical case, merely applying $(1, -1, 1, -1, \dots)$ to O is sufficient to determine the answer – the y^{th} pair $(1, -1)$ will be subject to a sign change, while all other elements will remain unchanged.

9 Conclusion

Our initial aim was to show that an electrical device capable of executing Deutsch's algorithm could be constructed. That we succeeded should not be altogether surprising – as we have emphasised, the algorithm merely consists of simple matrix/vector multiplication problem. Indeed, there are doubtless many other physical systems that could be used to implement the algorithm.

Since the Deutsch-Jozsa algorithm is an extension of the same idea, it is also unsurprising that an electrical implementation is possible. Here though the comparison becomes more interesting. In the quantum implementation, to double the size of the function's domain from 2^n values to 2^{n+1} requires one additional qubit. In our classical implementation, the number of electrical lines must also double. In both cases, the time of execution is constant, but the complexity of construction is exponentially worse in the electrical case.

It is often claimed that the remarkable property of the Deutsch and Deutsch-Jozsa algorithms is that multiple values of a function f are apparently evaluated at once. However, the complete description of f is contained in the oracle – f has already been evaluated. We can see this more clearly when we think of the oracle as a switchbox (see Figure 8), with a transposition for the i^{th} pair of wires whenever $f(i) = 1$ (for $i = 0, \dots, 2^n - 1$). It is then very easy to query multiple values of f at once and aggregate the results (Figure 9).

In the previous section, we turned our attention to Grover's algorithm, which to date has not appeared in the dequantisation literature. We observed that in a classical implementation, a single iteration of the Grover gate suffices to complete the search. Unlike our implementations of the Deutsch-Jozsa algorithm, this is a true reduction in the order of execution time – it becomes a fixed time operation – but at the same exponential cost in components and complexity of construction. Again, all this is possible because the oracle already contains the solution we seek, and without quantum-mechanical limitations, we are able to read out states directly.

Due to scaling restrictions, none of this is likely to have great practical application. We hope however that it will cast new light on a subtle question: "why are quantum computers faster?" We contend that the answer lies firstly in the capacity to perform certain matrix operations in constant time, and secondly in that only one qubit is required to double the size of the matrices concerned, rather than the more abstract notion of executing multiple oracle queries simultaneously. We have shown in particular that the Deutsch and Deutsch-Jozsa algorithms do not require quantum mechanics, they merely scale excellently when implemented on a quantum computer. Thus they should be celebrated not for doing something that cannot be done classically, but adapting to the restrictive rules of quantum mechanics and scaling beyond the capacity of digital computers.

A Op-amps

We present an overview of the fundamentals of operational amplifiers. We consider the properties of ideal electronic components, but touch briefly on practical limitations and considerations at the end of this appendix. For more details see [12] or similar texts.

As outlined in §6, we require two types of sub-circuits: summing gates and multiplicative gates. We can implement both of these with *operational amplifiers* (hereafter: *op-amps*). These are represented in circuit diagrams as follows – by convention we omit the power supply:

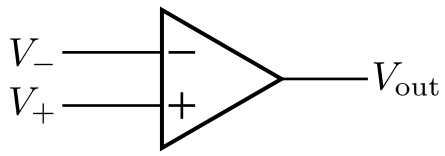

Figure 10

An op-amp is an electrical device that takes two input voltages, labelled V_+ and V_-, and produces an output voltage V_{out}, specified by

$$V_{\text{out}} = (V_+ - V_-)G, \tag{45}$$

where G is an extremely large real number, and is called the *open-loop gain* of the amplifier. The amplifier is supplied by power rails of voltage V_{sup_+} and V_{sup_-}. The output V_{out} is constrained to lie between these values – the amplifier is said to be *saturated* if V_{out} takes either extreme. We use the term *ground* to mean the constant voltage $(V_{\text{sup}_+} - V_{\text{sup}_-})/2$. Without any loss of generality, we take this quantity to be zero volts.

We make the following physical assumptions:

- G is so large that we may in all practical cases only consider the limit $G \to \infty$.
- The amplifier permits only a negligible current to flow through the input terminals V_+ and V_- – it merely measures the voltages and uses them to determine V_{out}.
- A reasonable amount of current may be drawn from the output terminal V_{out} without V_{out} changing appreciably.
- The voltages we deal with lie well within $(V_{\text{sup}_-}, V_{\text{sup}_+})$ – we may always rescale to ensure this is the case.

A.1 The inverting amplifier

At first glance it may appear that the device is only ever likely to emit $V_{\text{sup}_\pm}$. To make the device amplify a voltage linearly, we have to provide negative feedback: we siphon off some of the output current to change the voltage at the V_- terminal.

A design for a simple inverting amplifier is:

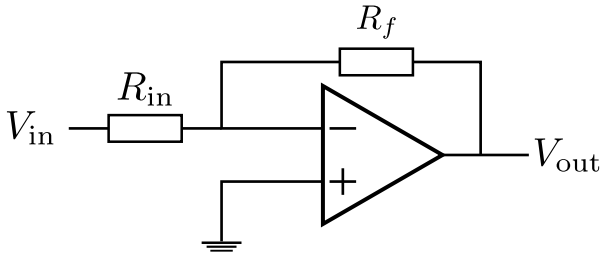

Figure 11

Note that the V_+ terminal is connected to ground, and thus plays no part in the amplification.

The voltage at the V_- pin is then determined by a potential divider between V_{out} and V_{in}. Let I be the current flowing from V_{out}. Then by applying Ohm's law twice:

$$(V_{\text{out}} - V_{\text{in}}) = I(R_f + R_{\text{in}}), \tag{46}$$
$$(V_- - V_{\text{in}}) = IR_{\text{in}}. \tag{47}$$

Taking the difference, we find $V_- = V_{\text{out}} - IR_f$ and substituting the top equation back in for I we conclude

$$V_- = \frac{R_{\text{in}}V_{\text{out}} + R_f V_{\text{in}}}{R_f + R_{\text{in}}}.$$

Substituting this into the gain equation and re-arranging, we find

$$V_{\text{out}}\left(1 - R_{\text{in}}\frac{G}{R_f + R_{\text{in}}}\right) = \underbrace{\frac{G}{R_f + R_{\text{in}}}}_{X} R_f V_{\text{in}}.$$

$$V_{\text{out}} = \frac{1}{X^{-1} - R_{\text{in}}} \cdot R_f V_{\text{in}},$$

$$V_{\text{out}} \approx -\frac{R_f}{R_{\text{in}}} V_{\text{in}}.$$

By careful choice of these two resistor values, this is enough to implement our attenuating gates for $\alpha < 0$, where α is the factor of attenuation. If we require $\alpha > 0$, we may simply use two inverting amplifiers in series.

A.2 Summing gates

A slight modification of the previous circuit may be used to produce a gate that takes a sequence of N voltages $V_1, \cdots, V_n$ and produces an output $V_{\text{out}} = -\sum V_i$:

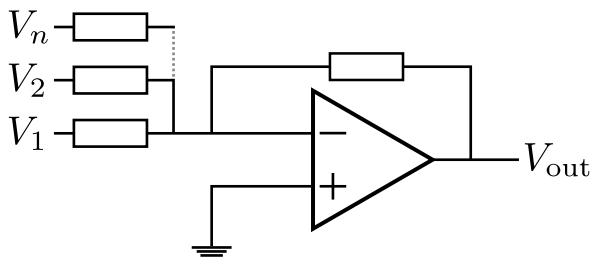

Figure 12

Note that all resistors here take the same value, which should be large to avoid dissipating too much heat.

By using N of these in parallel, we are able to implement the N rebit summing gate, albeit with an unwanted change of sign. We could of course use the previous circuit to right this, but since all lines will necessarily be negated, this is irrelevant to the computation and can merely be adjusted for in the interpretation of the output.

In fact, setting the input resistors to differing values R_i for $i = 1, \ldots, n$ and letting R_f denote the resistance of the remaining resistor, we can take a weighted sum

$$V_{\text{out}} = -R_f \sum_i \frac{V_i}{R_i}, \tag{48}$$

thus obviating the need for some of the attenuating gates.

A.3 Practicalities

A real op-amp does not quite live up to the idealisations in this discussion, but for all practical purposes if more than suffices and the above circuit designs may be assumed to be accurate to around 1%. It should be noted [12] that real op-amps are unable to produce V_{out} within a certain percentage of $V_{\text{sup}\pm}$, thus input voltages should be constrained to lie within a subinterval of $(V_{\text{sup}-}, V_{\text{sup}+})$.

B A prototype Hadamard gate

We present a simple prototype circuit for the implementation of *minus* the real Hadamard gate.

B.1 The power supply

We require three voltage *rails* – three lines from which one can draw current. We call these $V_{\text{sup}-}$, $V_{\text{sup}+}$ and 'ground'. We require that ground be the arithmetic mean of the other two. Such a supply is easily created by wiring two 9V batteries in series and taking the connection between the two as ground. Alternatively, a single power source may be split in two by a potential divider, and then buffered.[3]

[3]Buffering is a process by which one stabilises a voltage so that current may be drawn without loss of potential. In this case it may be achieved by connecting the output of the potential divider to the V_+ of an op-amp (an LM741 will do), then connecting the V_- to the V_{out} pins, and taking that wire as the buffered output.

B.2 The implementation

We divide the process in two: first we take the original two lines $[0|\psi]$ and $[1|\psi]$ and create a third: $-[1|\psi]$, by using an inverting amplifier as described above. We call this amplifier `I`.

The output $[0|\psi]'$ is then the attenuated sum of the two original wires, while the output $[1|\psi]'$ is the attenuated sum of the $[0|\psi]$ and $-[1|\psi]$ wires – in both cases we use summing amplifiers as described above. We use `P` to denote the former and `Q` to denote the latter summing amplifiers.

To effect the attenuation, weight the inputs to `P` and `Q` by careful choice of resistor value. We note that $\sqrt{2} \approx 47/33$. Conveniently, resistors of the value 47×10^n and 33×10^n Ohms are readily available for $n = 0, 1, 2, \ldots$.

We are now in a position to draw a suitable circuit diagram. Note that power supply requirements have been omitted for clarity.

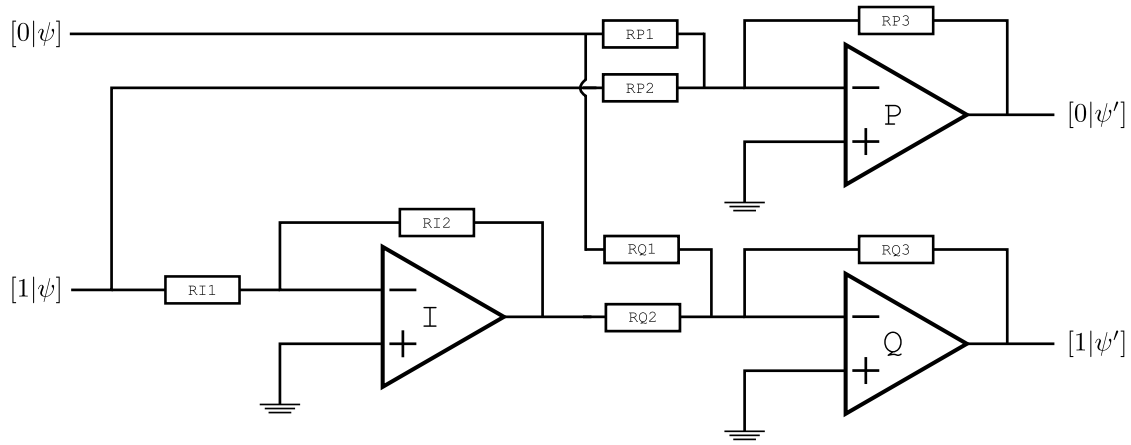

Figure 13

We specify the following components:

- $1 \times$ ST Microelectronics TSM104WIN quad op-amp, or similar.[4] Three LM741 op-amps would also do, but would increase clutter and effort.
- Resistors: `RP3` and `RQ3` = 33kΩ, all others 47kΩ.

C Experimental results

The circuit in Figure 13 was constructed on a breadboard, as shown in the diagram:

[4]The unused op-amp on this chip should have V_{out} connected to V_- and V_+ connected to ground to prevent it interfering with the operation of the remaining three [10].

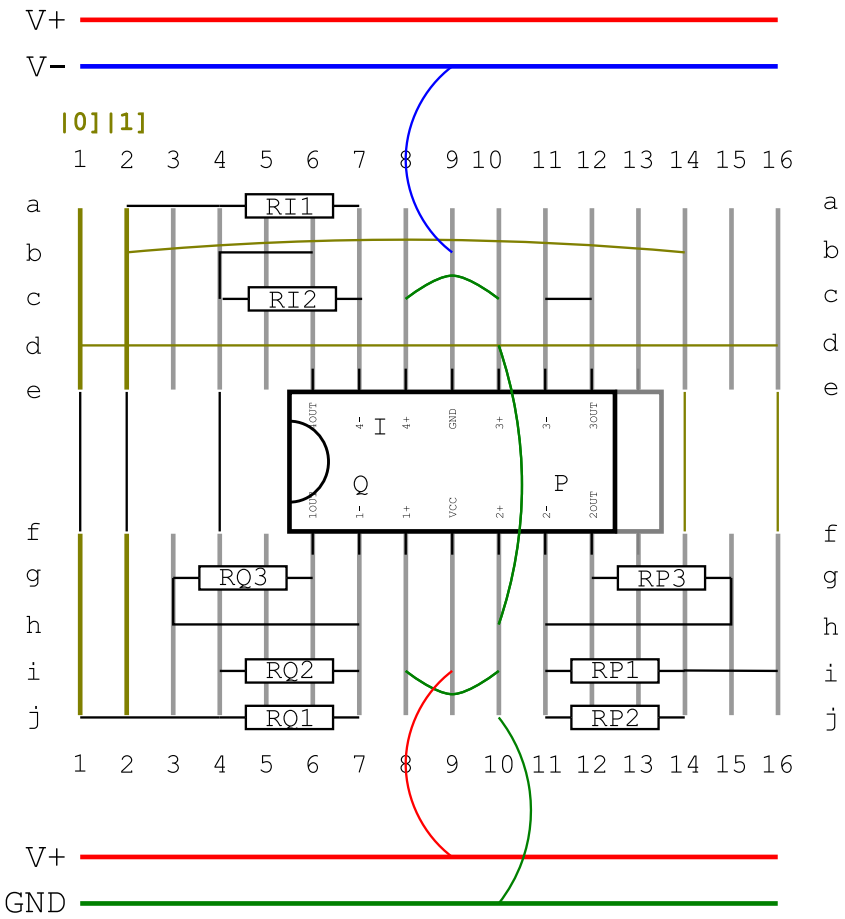

Figure 14: Implementation of a real $-H$ gate on a breadboard

It is worth noting that pins 7 and 8 on the integrated circuit relate to unused functionality, and are left open-circuit. A fourth op-amp on the chip, comprising pins 9, 10 and 11, was also unused, and was connected such as to avoid interference with the operation of the remaining three – see [10].

Finally, note that the pin `GND` on the integrated circuit refers to what we call $V_{\text{sup}-}$ – i.e. the negative supply, and not ground.

To power the circuit, we used a $9V$ battery to provide $V_{\text{sup}\pm}$. Ground, and two intermediate voltages slightly above and below ground, were provided by employing buffered potential dividers.

The following results were obtained in order:

No.	V_{Bat}	$\texttt{GND} - V_{\text{Sup}-}$	$V_{\text{Sup}+} - \texttt{GND}$	$\lvert 0\rangle_{\text{in}}$	$\lvert 1\rangle_{\text{in}}$	$\lvert 0\rangle_{\text{out}}$	$\lvert 1\rangle_{\text{out}}$	$\lvert 0\rangle_{\text{exp}}$	$\lvert 1\rangle_{\text{exp}}$
1	8.93	4.46	4.46	0.78	0.00	-0.54	-0.54	-0.55	-0.55
2	8.93	4.47	4.47	0.00	0.78	-0.54	0.55	-0.55	0.55
3	8.92	4.46	4.46	0.00	-0.77	0.54	-0.54	0.54	-0.54
4	8.91	4.45	4.45	-0.77	0.00	0.54	0.54	0.54	0.54
5	9.12	4.55	4.55	0.79	0.79	-1.11	0.00	-1.12	0.00
6	9.11	4.55	4.55	-0.78	-0.78	1.11	0.00	1.10	0.00
7	9.09	4.54	4.54	0.00	0.00	0.00	0.00	0.00	0.00
8	9.09	4.54	4.54	-0.78	0.79	0.00	1.11	-0.01	1.11
9	9.07	4.53	4.53	0.79	-0.78	0.00	-1.10	-0.01	-1.11

All figures are given in volts and the columns have the following meanings:

- V_{Bat}: The voltage across the terminals of the battery.
- $\texttt{GND} - V_{\text{Sup}-}$: The potential difference between `GND` (ground) and the negative terminal of the battery.
- $V_{\text{Sup}+} - \texttt{GND}$: The potential difference between the positive terminal of the battery and `GND`.
- $\lvert 0\rangle_{\text{in}}$ and $\lvert 1\rangle_{\text{in}}$: The voltages applied as inputs to the circuit, representing $[\psi|0]$ and $[\psi|1]$ respectively.
- $\lvert 0\rangle_{\text{out}}$ and $\lvert 1\rangle_{\text{out}}$: The voltages measured at the outputs of the circuit – i.e. the experimental (measured) values of $-H\psi$.
- $\lvert 0\rangle_{\text{exp}}$ and $\lvert 1\rangle_{\text{exp}}$: The expected values of the two components of $-H\,|\psi]$.

As no appreciable current is drawn during the course of the circuit's operation, slight changes in the voltage of the battery are attributable to changes in ambient temperature. There was a short delay between results 4 and 5, which is consistent with this theory. All voltages were measured using a standard digital multimeter, with a displayed precision of $\pm 0.01V$. We thus note that every single result fell within the tolerance of the measurement equipment. Assuming the measurements are as accurate as they are precise, we can put the circuit's overall tolerance at within 1%.

As discussed in §6.2, implementing Deutsch's algorithm will require six of these connected appropriately. This is an easy exercise left to the reader.

References

[1] A. A. Abbott. The deutsch-jozsa problem: De-quantisation and entanglement. 2009. http://arxiv.org/abs/0910.1990v3.

[2] A. A. Abbott and C. S. Calude. Understanding the quantum computational speed-up via de-quantisation. In *Developments in Computational Models*, 2010. http://arxiv.org/abs/1006.1419.

[3] C. S. Calude. De-quantizing the solution of Deutsch's problem. *International Journal of Quantum Information*, 5(3):409–415, June 2007. http://arxiv.org/abs/quant-ph/0610220.

[4] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. R. Soc. Lond. A*, 400(1818):97–117, July 1985.

[5] D. Deutsch and R. Jozsa. Rapid solutions of problems by quantum computation. *Proc. R. Soc. Lond. A*, 439(1907):553–558, December 1992.

[6] L. K. Grover. A fast quantum mechanical algorithm for database search. May 1996. http://arxiv.org/abs/quant-ph/9605043.

[7] J. A. Jones and M. Mosca. Implementation of a quantum algorithm on a nuclear magnetic resonance quantum computer. *J. Chem. Phys.*, 109(1648), 1998.

[8] O. Kaan. Deutsch algorithm on classical circuits. 2008. http://arxiv.org/abs/0803.3183.

[9] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[10] J. Posvic. What shall we do with the unused op-amp? http://www.scribd.com/doc/20697674/What-shall-we-do-with-a-unused-OP-Amp.

[11] T. Rudolph and L. K. Grover. A 2 rebit gate universal for quantum computing. October 2002. http://arxiv.org/abs/quant-ph/0210187.

[12] J. Watson. *Mastering Electronics*. Macmillan Press Ltd, fourth edition, 1996.