Quantum circuits for new applications of the Deutsch-Jozsa algorithm

José Manuel Bravo

pestecnologia@gmail.com

Prof. Computer Science and Technology, High School, Málaga, Spain

Abstract

In this paper a serial of new applications on binary strings based on the Deutsch-Jozsa algorithm are presented. These new applications can be used to calculate the Hamming weight and the Hamming distance of a binary string or to compare if two binary strings have the same Hamming weight or the same Hamming distance. To test the applications the quantum computer prototype that IBM has given open access to on the cloud has been used to test the results.

Keywords: quantum algorithm; Hamming weight; Hamming distance; Deutsch-Jozsa algorithm

1. Introduction

The advantage of the quantum computation to solve more efficiently than classical computation some algorithms is one of the most important key for scientists to continue researching and developing new quantum algorithms. Grover's searching quantum algorithm is able to select one entry between 2^n with output 1 in only $O(\sqrt{2^n})$ [1] queries to an oracle. Shor's factoring quantum algorithms [2] is able to factorize one number faster than any classical algorithms known.

The Hamming weight of two binary strings (messages) of equal length is the number of symbols different of the zero-symbol used in the alphabet. It represents the number of 1's in the string or the digit sum of the binary representation. The Hamming weight is widely used in different disciplines, including information theory, coding theory, and cryptography, and his efficient implementation has been widely studied [3][4][5].

The Hamming distance in information theory between two strings is the minimum number of substitutions required to change one string into the other. It is used in coding theory to define the error detecting and the error correcting codes. The running time of classical algorithms is proportional to the Hamming distance or to the number of bits in the inputs.

These quantum algorithms uses quantum parallelism as the fundamental feature to evaluate a function implemented in a quantum oracle for many different values of x at the same time because it exploits the ability of a quantum computer to be in superposition of different states.

Moreover it uses a property of quantum mechanics known as interference. In a quantum computer it is possible for the two alternatives to interfere with one another to yield some global property of a function f , by using the Hadamard gate to recombine the different alternatives.

The essence of the design of this algorithm is the choice of a function and a final transformation that allows efficient determination of a property of the binary string.

The quantum algorithm is based on the Deutsch-Jozsa algorithm that is able to determinate if a function is constant or balanced using only one query to a quantum oracle [6]. The novelty of these new applications is to encode two binary strings before apply the Deutsch-Jozsa algorithm; the first one encode the string or strings for which we want to calculate some property, and the other one acts like an ancilla string. The ancilla string is one of the key because different ancillas strings are applied to different applications. Coding in that way allow us to calculate the Hamming weight of a binary string, to determinate the Hamming distance of a binary string, to determinate if two binary strings have the same Hamming weight or to determinate if two binary strings have the same Hamming distance. Theoretically only one query to the oracle is needed in all the applications.

IBM releases in 2016 a 5-qubit universal quantum computer prototype accessible on the cloud. To test the algorithm, two experiments have been simulated on the IBM Q Experience composer.

2. The Deutsch-Jozsa algorithm

The goal is to find an algorithm which tells if a function f is constant or balanced with the least possible number of evaluations of an oracle.

The algorithm is divided into the following steps:

1. Prepare an $(n+1)$ -qubit register in the state $|\Psi_0\rangle = |0\rangle^{\otimes n} \otimes |1\rangle$. First n qubits work as input qubits of the function, while the $(n+1)$ st is an ancilla qubit to store temporary information.
2. Apply the Hadamard transformation to the register. Then we have the state:

$$|\Psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

3. Perform a quantum transformation Bf , associated to a function f defined by $Bf|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$ where $f(x)$ is any function $f: \{0,1\}^n \rightarrow \{0,1\}$. Then we obtain the following state.

$$|\Psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

4. Apply a Hadamard transformation on the first n qubit.

$$|\Psi_3\rangle = \frac{1}{2^n} \sum_{x,y} (-1)^{f(x)+x \cdot y} |y\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

$$x \cdot y = x_{n-1}y_{n-1} \oplus x_{n-2}y_{n-2} \oplus \dots \oplus x_0y_0$$

5. The first n qubits are measured. Following the algorithm if the probability that the measurements all give outcome 0, is 1, the function f is constant and balanced is that probability is 0.

3. Calculating the Hamming weight

The algorithm to calculate the Hamming weight is based on the Deutsch-Jozsa algorithm so the steps are the same described above.

The function f associated to the transformation Bf is the following. Suppose we have to calculate the Hamming weight of a binary string of length k . In that case we define a function from $n = \log_2(k) + 1$ bits to 1 bit in the following way:

- On the first 2^{n-1} inputs of the function we encode the binary string (message) based on the position of each bit on the string (message).
- On the last 2^{n-1} inputs of the function we fix the binary string with all zero's.

Suppose we want to calculate the Hamming weight of the string $\mathbf{a} = \mathbf{10}$. Then function f will be:

$$f: \{0,1\}^{\log_2(2)+1} \rightarrow \{0,1\}$$

Input $ x\rangle$	$F(x)$
00	0
01	1
10	0
11	0

The transformation Bf :

$$Bf|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$$

has associated a unitary permutation so it can be implemented in a reversible quantum circuit.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Then we apply the Bf one time and the Hadamard transformations. After the n qubits are measured, we can observe:

$$hw(a) = 2^{n-1} \cdot (1 - \sqrt{P|0\rangle}) \quad (1)$$

$$hw(a) = 2^{n-1} \cdot \sqrt{P|2^{n-1}\rangle} \quad (2)$$

where $P|2^{n-1}\rangle$ is the probability of measure the state $|2^{n-1}\rangle$ and $P|0\rangle$ is the probability of measure the state $|0\rangle$ for any number n . As you can see if the Hamming weight tell us the number of symbols different of the zero-symbol used in the alphabet, the number $2^{n-1} \cdot (1 - \sqrt{P|0\rangle})$ tell us the number of symbols equal to the zero-symbol in the string.

Proof

If we want to calculate the Hamming weight of a binary string s of length k ($n = \log_2(k) + 1$), we define m_L like the first 2^{n-1} bits and m_H the last 2^{n-1} bits of the message:

$$m = m_{2^{n-1}-1} \dots m_0 = m_H \& m_L$$

$$m_H = m_{2^{n-1}-1} \dots m_{2^{n-1}}$$

$$m_L = m_{2^{n-1}-1} \dots m_0 = s_k \dots s_1$$

Following the algorithm m_H will be the binary string with all zero's. Now consider the summation

$$\sum_x (-1)^{f(x)+x \cdot y} |y\rangle \quad (3)$$

for the state $|0\rangle$ and $|2^{n-1}\rangle$ in m_H :

$$\sum_{x \in m_H} (-1)^{0+x \cdot (0 \dots 0)} |0\rangle = 2^{n-1} \quad (4)$$

$$\sum_{x \in m_H} (-1)^{0+x \cdot (10 \dots 0)} |2^{n-1}\rangle = -2^{n-1} \quad (5)$$

Now if we consider the summation (3) for the state $|0\rangle$ and $|2^{n-1}\rangle$ in m_L we will obtain the same result:

$$\begin{aligned} \sum_{x \in m_L} (-1)^{f(x)+x \cdot (0 \dots 0)} |0\rangle &= \\ &= 2^{n-1} - (2 \cdot hw(m_L)) \end{aligned} \quad (6)$$

$$\sum_{x \in m_L} (-1)^{f(x)+x \cdot (10 \dots 0)} |2^{n-1}\rangle = \quad (7)$$

$$= 2^{n-1} - (2 \cdot hw(m_L))$$

where $hw(m_L)$ is the Hamming weight of the substring m_L . After that we consider the contribution of (4) and (6) to the amplitude associated with the state $|0\rangle$ and we will obtain

$$\frac{1}{2^n} \cdot (2^{n-1} - 2 \cdot hw(m_L) + 2^{n-1}) = \quad (8)$$

$$= 1 - \frac{hw(m_L)}{2^{n-1}}$$

If we consider the contribution of (5) and (7) to the amplitude associated with the state $|2^{n-1}\rangle$, we will obtain

$$\frac{1}{2^n} \cdot (2^{n-1} - 2 \cdot hw(m_L) - 2^{n-1}) = \quad (9)$$

$$= \frac{-hw(m_L)}{2^{n-1}}$$

when the state $|0\rangle$ is measured we will obtain:

$$P|0\rangle = \left(1 - \frac{hw(m_L)}{2^{n-1}}\right)^2 \quad (8)$$

from where we can easily obtain (1). And if the state $|2^{n-1}\rangle$ is measured we will obtain:

$$P|0\rangle = \left(\frac{-hw(m_L)}{2^{n-1}}\right)^2 \quad (10)$$

from where we can easily obtain (2).

4. Calculating the Hamming distance

Suppose we have to calculate the Hamming distance of two binary strings A y B of length k . The steps are the following:

1. Calculate $C = A \oplus B$
2. Build the function f

$$f: \{0,1\}^{n=\log_2(k)+1} \rightarrow \{0,1\}$$

that encodes the binary string C in the first $n/2$ inputs and the binary string *all zero's* in the last $n/2$ inputs (ancilla string).

3. Prepare an $(n+1)$ -qubit register in the state $|\Psi_0\rangle = |0\rangle^{\otimes n} \otimes |1\rangle$. First n qubits work as input qubits of the function, while the $(n+1)$ st qubit is an ancilla qubit to store temporary information.

4. Apply the steps 2, 3, 4 and 5 of the Deutsch-Jozsa algorithm.
5. The states $|2^{n-1}\rangle$ and $|0\rangle$ are measured:

$$hd(A,B) = hw(C) = 2^{n-1} \cdot \sqrt{P|2^{n-1}\rangle}$$

$$hd(A,B) = hw(C) = 2^{n-1} \cdot (1 - \sqrt{P|0\rangle})$$

Below you can see the probabilities obtained for the Hamming weight (hw) for different n values.

hw	$P 0\rangle$	$P 2^{n-1}\rangle$
0	1,00	0,00
1	0,25	0,25
2	0,00	1,00

Table 1 $n=1$

hw	$P 0\rangle$	$P 2^{n-1}\rangle$
0	1,00000	0,00000
1	0,56250	0,06250
2	0,25000	0,25000
3	0,06250	0,56250
4	0,00000	1,00000

Table 1 $n=2$

hw	$P 0\rangle$	$P 2^{n-1}\rangle$	hw	$P 0\rangle$	$P 2^{n-1}\rangle$
0	1,00000	0,00000	5	0,14062	0,39062
1	0,76562	0,01562	6	0,06250	0,56250
2	0,56250	0,06250	7	0,01562	0,76562
3	0,39062	0,14062	8	0,00000	1,00000
4	0,25000	0,25000			

Table 2 $n=3$

hw	$P 0\rangle$	$P 2^{n-1}\rangle$	hw	$P 0\rangle$	$P 2^{n-1}\rangle$
0	1,00000	0,00000	9	0,19141	0,31641
1	0,87891	0,00391	10	0,14063	0,39063
2	0,76563	0,01563	11	0,09766	0,47266
3	0,66016	0,03516	12	0,06250	0,56250
4	0,56250	0,06250	13	0,03516	0,66016
5	0,47266	0,09766	14	0,01563	0,76563
6	0,39063	0,14063	15	0,00391	0,87891
7	0,31641	0,19141	16	0,00000	1,00000
8	0,25000	0,25000			

Table 3 $n=4$

5. Comparing the Hamming weight of two binary strings

Suppose that we are given two binary strings a, b of equal length k . The goal is to determinate if both strings have the same Hamming weight. Classically k queries are necessary and sufficient to solve the problem. Using this quantum algorithm, one query will be sufficient to solve it.

The steps are the following:

1. Build the function f

$$f: \{0,1\}^{n=\log_2(k)+2} \rightarrow \{0,1\}$$

that encodes the binary string a in the first $n/4$ inputs, the binary string b in the next $n/4$ inputs and the following ancilla string: first $n/4$ bits all one's next $n/4$ bits all zero's.

2. Prepare an $(n+1)$ -qubit register in the state $|\Psi_0\rangle = |0\rangle^{\otimes n} \otimes |1\rangle$. First n qubits work as input qubits of the function, while the $(n+1)$ st qubit is an ancilla qubit to store temporary information.
3. Apply the steps 2, 3, 4 and 5 of the Deutsch-Jozsa algorithm.
4. The states $|2^{n-2}\rangle$ and $|2^{n-2} + 2^{n-1}\rangle$ are measured:

$$\begin{aligned} P|2^{n-2}\rangle &= P|2^{n-2} + 2^{n-1}\rangle = \\ &= \begin{cases} 0.25 & \text{if } hw(a) = hw(b) \\ \text{otherwise} & \text{if } hw(a) \neq hw(b) \end{cases} \end{aligned}$$

Proof

If we want to determinate if two binary strings a, b of equal length k have the same Hamming weight ($n = \log_2(k) + 2$), we define m_L like the first 2^{n-1} bits and m_H the last 2^{n-1} bits of the message:

$$\begin{aligned} m &= m_{2^{n-1}-1} \dots m_0 = m_H \& m_L \\ m_H &= m_{2^{n-1}-1} \dots m_{2^{n-1}} \\ m_L &= a \& b = a_k \dots a_1 b_k \dots b_1 \end{aligned}$$

Following the algorithm m_H will be ancilla string: $n/4$ bits all one's and $n/4$ bits all zero's. Now consider the summation (3) for the state $|2^{n-2}\rangle$ and $|2^{n-2} + 2^{n-1}\rangle$ in m_H :

$$\sum_{x \in m_H} (-1)^{f(x)+x \cdot (010 \dots 0)} |2^{n-2}\rangle = -2^{n-1} \quad (11)$$

$$\begin{aligned} \sum_{x \in m_H} (-1)^{f(x)+x \cdot (110 \dots 0)} |2^{n-2} + 2^{n-1}\rangle \\ = 2^{n-1} \end{aligned} \quad (12)$$

Now if we consider again the summation (3) for the state $|2^{n-2}\rangle$ and $|2^{n-2} + 2^{n-1}\rangle$ in m_L we will obtain

$$\sum_{x \in m_L} (-1)^{f(x)+x \cdot (01 \dots 0)} |2^{n-2}\rangle = 0 \quad (13)$$

$$\sum_{x \in m_L} (-1)^{f(x)+x \cdot (11 \dots 0)} |2^{n-2} + 2^{n-1}\rangle = 0 \quad (14)$$

only if the Hamming weight of binary strings a, b are equals because the coefficients vanishes for half string being a and the other half b with the same numbers of zero's and one's. Otherwise these summation will not be zero.

If we consider the contribution of (11) and (13) to the amplitude associated with the state $|2^{n-2}\rangle$, we will obtain

$$\frac{1}{2^n} \cdot (-2^{n-1}) = -\frac{1}{2} \text{ if } hw(a) = hw(b) \quad (15)$$

And if we consider the contribution of (12) and (14) to the amplitude associated with the state $|2^{n-2} + 2^{n-1}\rangle$

$$\frac{1}{2^n} \cdot (2^{n-1}) = \frac{1}{2} \text{ if } hw(a) = hw(b) \quad (16)$$

When the state $|2^{n-2}\rangle$ and $|2^{n-2} + 2^{n-1}\rangle$ are measured we will obtain:

$$\begin{aligned} P|2^{n-2}\rangle &= P|2^{n-2} + 2^{n-1}\rangle = 0.25 \\ &\text{if } hw(a) = hw(b) \end{aligned} \quad (17)$$

Moreover if $hd(a, b) = 0$ (Hamming distance) we will obtain all outcomes 0 when we measure the following states:

$$|2^{n-2} + 1\rangle \dots |2^{n-1} - 1\rangle \quad (18)$$

$$|2^{n-2} + 2^{n-1} + 1\rangle \dots |2^n - 1\rangle \quad (19)$$

6. Working with the ancillas strings

Other interesting application is the following: the goal is to determinate which strings have an odd Hamming weight and which ones have an even Hamming weight in one query to an oracle.

This problem can be solved with this method encoding the following n bits ancilla string:

$$1000 \dots 0001$$

(See experiment 5)

7. Simulations on IBM Quantum Experience

Experiment 1 (figure 1): calculate de Hamming distance of the following strings:

$$a = 10100101 \quad b = 01100101$$

In this case $n=4$. The first step is to calculate

$$c = a \oplus b = 11000000$$

and to define the function f

Input ($q[3], q[2], q[1], q[0]$)	f	Input ($q[3], q[2], q[1], q[0]$)	f
0000	0	1000	0
0001	0	1001	0
0010	0	1010	0
0011	0	1011	0
0100	0	1100	0
0101	0	1101	0
0110	1	1110	0
0111	1	1111	0

Table 4 ($f = q_1 \wedge q_2 \wedge \overline{q_3}$)

The results we obtain after 8192 shots are:

State	Probability
0⟩	0,566
2⟩	0,062
4⟩	0,066
6⟩	0,061
8⟩	0,063
10⟩	0,058
12⟩	0,061
14⟩	0,064

Table 5

$$hd(a, b) = hw(c) = \lfloor \sqrt{0,063} \cdot 2^3 \rfloor = 2$$

$$hd(a, b) = hw(c) = 2^3 - \lfloor \sqrt{0,566} \cdot 2^3 \rfloor = 2$$

Experiment 2 (figure 2): calculate de Hamming distance of the following strings:

$$a = 1001 \quad b = 0001$$

In this case $n=3$. The first step is to calculate

$$c = a \oplus b = 1000$$

and to define the function f

Input ($q[2], q[1], q[0]$)	f	Input ($q[2], q[1], q[0]$)	f
000	0	100	0
001	0	101	0
010	0	110	0
011	1	111	0

Table 6 ($f = q_0 \wedge q_1 \wedge \overline{q_2}$)

The results we obtain after 8192 shots are:

State	Probability
0⟩	0,565
1⟩	0,066
2⟩	0,062
3⟩	0,052
4⟩	0,066
5⟩	0,065
6⟩	0,059
7⟩	0,065

Table 7

$$hd(a, b) = hw(c) = \lfloor \sqrt{0,066} \cdot 2^2 \rfloor = 1$$

$$hd(a, b) = hw(c) = 2^2 - \lfloor \sqrt{0,565} \cdot 2^2 \rfloor = 1$$

Figure 1 Quantum circuit to calculate de Hamming distance of $a = 10100101$ and $b = 01100101$

Figure 2 Quantum circuit to calculate de Hamming distance of $a = 1001$ and $b = 0001$

Experiment 3 (figure 3 and 4): determinate if the binary strings a, b have the same Hamming weight

$$a = 1010 \quad b = 1010$$

In this case $n=4$ and the function f will be:

Input ($q[3], q[2], q[1], q[0]$)	f	Input ($q[3], q[2], q[1], q[0]$)	f
0000	1 (a_4)	1000	0
0001	0 (a_3)	1001	0
0010	1 (a_2)	1010	0
0011	0 (a_1)	1011	0
0100	1 (b_4)	1100	1
0101	0 (b_3)	1101	1
0110	1 (b_2)	1110	1
0111	0 (b_1)	1111	1

Table 8 ($f = (q_2 \wedge q_3) \vee (\overline{q_0} \wedge \overline{q_3})$)

The results we obtain after 8192 shots that you can see below on figure 4, are:

State	Probability
$ 1\rangle$	0,248
$ 4\rangle$	0,24
$ 9\rangle$	0,25
$ 12\rangle$	0,262

Table 9

As we can see the probability of the states $|4\rangle$ and $|12\rangle$ are approximately 0.25 and the outcomes of the states $|5\rangle, |6\rangle, |7\rangle, |13\rangle, |14\rangle, |15\rangle$ are all zero's (see (18) and (19)).

Figure 3 Quantum circuit to compare de Hamming weight of $a = 1010$ and $b = 1010$

Figure 4 Quantum results for the circuit on figure 3

Experiment 4 (figure 5 and 6): determinate if the binary strings a, b have the same Hamming weight

$$a = 0000 \quad b = 0011$$

In this case $n=4$ and the function f will be:

Input ($q[3], q[2], q[1], q[0]$)	f	Input ($q[3], q[2], q[1], q[0]$)	f
0000	0 (a_4)	1000	0
0001	0 (a_3)	1001	0
0010	0 (a_2)	1010	0
0011	0 (a_1)	1011	0
0100	0 (b_4)	1100	1
0101	0 (b_3)	1101	1
0110	1 (b_2)	1110	1
0111	1 (b_1)	1111	1

Table 10 ($f = (q_2 \wedge q_3) \vee (q_1 \wedge q_2)$)

The results we obtain after 8192 shots that you can see below on figure 4, are:

State	Probability
$ 0\rangle$	0,063
$ 2\rangle$	0,057
$ 4\rangle$	0,566
$ 6\rangle$	0,059
$ 8\rangle$	0,068
$ 10\rangle$	0,063
$ 12\rangle$	0,059
$ 14\rangle$	0,066

Table 11

As we can see the probability of the states $|4\rangle$ and $|12\rangle$ are distinct of 0.25

Figure 5 Quantum circuit to compare de Hamming weight of $a = 0000$ and $b = 0011$

Figure 6 Quantum results for the circuit on figure 5

Experiment 5 (figure 7 and 8): determinate which of the outcomes of a quantum circuit with 3 qbits represents in binary code a string with an odd Hamming weight and which one with an even Hamming weight.

The function f will be:

Input ($q[2], q[1], q[0]$)	f
000	1
001	0
010	0
011	0
100	0
101	0
110	0
111	1

Table 10 ($f = (q_2 \wedge q_1 \wedge q_0) \vee (\overline{q_2} \wedge \overline{q_1} \wedge \overline{q_0})$)

State	Probability
$ 0\rangle$ (000) ($hw = 0$)	0,247
$ 1\rangle$ (001) ($hw = 1$)	0
$ 2\rangle$ (010) ($hw = 1$)	0
$ 3\rangle$ (011) ($hw = 2$)	0,252
$ 4\rangle$ (100) ($hw = 1$)	0
$ 5\rangle$ (101) ($hw = 2$)	0,254
$ 6\rangle$ (110) ($hw = 2$)	0,247
$ 7\rangle$ (111) ($hw = 3$)	0

Table 12

In this case the algorithm works better than any classical one because in just only one query to an oracle, the quantum circuit shows the even Hamming weight outcomes with probability greater than zero and zero for odd Hamming weight outcomes. This ancilla string works for any $n > 2$.

The results we obtain after 8192 shots are showed below on figure 7.

Figure 7 Quantum circuit to determinate odd and even Hamming weight outcomes

Figure 7 Quantum results for the circuit on figure 7

8. Conclusions

Although these new applications of the Deutsch-Jozsa algorithm works on the simulator of IBM, the reality is that we have a great constraint because the low probabilities of the applications. Comparing if two binary strings have the same Hamming weight or the same Hamming distance theoretically only need one query to a quantum oracle to solve it, but only if the measurements at the end of the process give us clearly an expected value in the outcomes, otherwise we will have to query more times. Anyway it is a new approach to the Deutsch-Jozsa algorithm and maybe not the last one. Running these applications on the ibmqx4 or ibmqx2 quantum computer and studying the results and errors would be the next step and of course testing with others ancillas to study if another string or function property is still hidden.

Data availability

All relevant data are available within the paper.

Competing interests

The author confirms that there are no known conflicts of interest associated with this publication and there has been no financial support for this work that could have influenced its outcome. The author declares no competing interests.

References

- [1] Lov K. Grover (1996). A fast quantum mechanical algorithm for database search (Bell Labs, Murray Hill NJ).
- [2] Peter W. Shor (1995). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer (AT&T Research)
- [3] Muła, Wojciech; Kurz, Nathan; Lemire, Daniel (January 2018), "Faster Population Counts Using AVX2 Instructions", *Computer Journal*, 61 (1), doi:10.1093/comjnl/bxx046
- [4] Knuth, Donald Ervin (2009). "Bitwise tricks & techniques; Binary Decision Diagrams". *The Art of Computer Programming*. Volume 4, Fascicle 1. Addison–Wesley Professional. ISBN 0-321-58050-8. (NB. Draft of Fascicle 1b available for download.)
- [5] Hewlett-Packard HP-16C Computer Scientist Owner's Handbook (PDF). Hewlett-Packard Company. April 1982. 00016-90001. Archived (PDF) from the original on 2017-03-28. Retrieved 2017-03-28
- [6] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London A*, 439:553–558, Jan 1992.
- [7] Quantum information science. Michael Nielsen and Isaac Chuang. Cambridge University Press. ISBN 978-1-107-00217-3
- [8] Quantum Computing. From Linear Algebra to Physical Realizations. Mikio Nakahara and Tetsuo Ohmi. CRC Press. ISBN 978-0-7503-0983-7