QUANTUM COMPUTERS - EXAMPLE OF EXECUTING THE DEUTSCH QUANTUM ALGORITHM USING GOOGLE COLAB

Željko Eremić*,
[0009-0003-3310-3081]

Tanja Sekulić,
[0000-0002-0977-4964]

Iris Borjanović Trusina
[0009-0001-5276-338X]

Technical College of
Applied Sciences in Zrenjanin,
Zrenjanin, Serbia

Abstract:

In this paper, an introduction to the basic principles of quantum computing is provided, from concepts of quantum mechanics to their mathematical and computational representation. The core phenomena of quantum mechanics, such as superposition, interference, and entanglement, are explained, and their applications in quantum information processing are described.

Mathematical representation of quantum information, starting from qubits, quantum gates, and quantum operations, is presented and explained. Also, the overview of basic quantum algorithms is given and compared to classical approaches. The quantum Deutsch algorithm is presented and discussed mathematically.

Prerequisites and programming code in Python for executing the quantum Deutsch's algorithm are presented. Figures of quantum circuits and execution results are given. Finally, the results of the execution of this algorithm are discussed.

Keywords:

Quantum Mechanics, Quantum Computers, Quantum Operations, Deutsch Algorithm.

INTRODUCTION

Quantum mechanics is a mathematical framework that extraordinarily well describes the subatomic world and has great experimental confirmation. Its ideas and postulates are not always obvious and can be counterintuitive with respect to people's everyday experiences in the macroscopic world. Quantum mechanics' applications in everyday life are immense. For example, all electronics, modern medicine diagnostic equipment, and nuclear power plants were developed thanks to its use.

The first ideas about quantum computers [1] and the application of quantum principles [2] to data processing appeared more than fifty years ago. In the meantime, this idea has successfully come to fruition, and we now have practical realizations of quantum computers. Today, this field of research is in a phase of great expansion. The advantage of quantum computers over classical ones is that, in certain cases, they can be much faster.

Correspondence:
Željko Eremić

e-mail:
zeljko.eremic@vts-zr.edu.rs

DOI: 10.15308/Sinteza-2026-334-339

Main fields of application include cryptography, simulation of quantum systems, optimizations, the production of pharmaceuticals, and financial modelling. However, they are not ideal for dealing with a large amount of input data simultaneously (big data). Many countries and leading companies (Google, IBM, Microsoft, Amazon, and others) are investing a significant amount of money, time, and effort into quantum computing research with the desire to achieve supremacy in the field.

2.. THEORETICAL FRAMEWORK

Quantum computers are based on the principles of quantum mechanics and supported by a powerful mathematical framework [3].

2.1. QUANTUM MECHANICS

The main difference between classical and quantum computers is that while classical computers use bits, quantum ones use quantum bits, or qubits. Classical bits can be in a state of 0 or 1, while qubits can also exist as a superposition of these two states as long as they are not measured. Upon measurement, with a certain probability, they can only be in state 0 or 1. Sometimes an analogy is made with a rotating coin: until the coin stops, it is simultaneously heads and tails. This superposition of states allows quantum computers to perform many processes in parallel, making them faster than classical computers in certain situations.

Another important quantum mechanical principle used in quantum computers is entanglement. When we have a pair of entangled electrons, for example, they are both in a superposition of spin-up and spin-down states before measurement. If we measure the spin of one electron, it becomes either spin-up or spin-down, and we automatically instantaneously know the spin of the other electron without measuring it, independent of how far apart these two electrons are. This property is used in quantum computers where a large number of qubits are entangled.

The principle of interference is used in decision-making. When two superimposed probability waves have positive interference and maximum peaks, those answers are the most probable. On the contrary, negative interference (where two waves cancel each other out) leads to answers that are filtered out.

Interaction with the environment, called decoherence, is an unwanted phenomenon that causes a quantum system to collapse into a classical one. The goal of hardware designers is to minimize decoherence. For example, superconducting quantum computers operate at very low temperatures (a few millikelvins) to reduce this effect.

There exist many different technologies used for the realization of quantum computers. Small superconducting circuits, electrons, trapped ions, photons, or neutral atoms can all be used as bits. Classification can also be based on the computing model, such as gate-based or analogue quantum computing models. This paper is based on an analysis performed on the Google Colab platform, which is an environment for writing, simulating, and performing quantum algorithms.

2.2. MATHEMATICAL REPRESENTATION OF QUANTUM STATES AND OPERATIONS

In quantum computing, there are qubits, as the basic carriers of information, and they represent the quantum states. Mathematically, qubit is represented as: $|\psi\rangle = a|0\rangle + b|1\rangle$ or in vector form: $|\psi\rangle = \begin{pmatrix} a \\ b \end{pmatrix}$, where $|a|^2 + |b|^2 = 1$, and a, b∈C. The coefficients a, b∈C define amplitudes, whose quadratic value represents the probability of measuring in base $|0\rangle$ or $|1\rangle$. The classical states are 0 and 1, represented as follows: $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Every qubit is a superposition of classical states 0 and 1.

Every state of the single qubit can be represented graphically as a point on the surface of the sphere. There is a sphere, called the Bloch sphere, with radius $r = 1$, and there are angles: θ ($0 \le \theta \le \pi$), which controls latitude (amplitude, and φ ($0 \le \varphi < 2\pi$), which controls longitude (phase). Basic states are on the poles of the Bloch sphere, 0 on the north pole and 1 on the south pole, Figure 1.

Operations in quantum computing are called quantum gates, and are defined as multiplying a qubit's vector ψ represented as $|\psi\rangle$ by an operation matrix U, $|\psi'\rangle = U|\psi\rangle$. The matrix U must be unitary, $UU^\dagger = U^\dagger U = I$, where $U^\dagger$ marks the conjugate transpose of the matrix U. This means that every unitary matrix has its inverse matrix ($U^{-1} = U^\dagger$), which is particularly important because it ensures that every quantum operation can be reversed. Basically, quantum operations rotate the qubit's vector along the surface of the Bloch sphere; it does not change its length, it only changes its amplitudes and phases. Superposition, interference, and entanglement are defined by quantum gates. There are universal quantum gates, which denote basic quantum operations and enable the design of a quantum algorithm. There are a variety of quantum gates, but for this paper, only three of them will be presented.

Pauli X-gate is a quantum gate that corresponds to classical NOT. It changes $|0\rangle \leftrightarrow |1\rangle$ and $|1\rangle \leftrightarrow |0\rangle$. On the Bloch sphere, it rotates the state vector by angle π around the x-axis. The matrix of Pauli X-gate is $X=\left(\begin{smallmatrix}0 & 1\\ 1 & 0\end{smallmatrix}\right)$. If Pauli X-gate is applied onto state $|0\rangle$, the opposite state $|1\rangle$ is obtained.

The Hadamard gate creates superposition by rotating the state vector from the z-axis onto the x-axis. The matrix of the Hadamard gate is $\mathrm{H} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$. When the Hadamard gate is applied to state $|0\rangle$ the superposition of states $|0\rangle$ and $|1\rangle$ is obtained, meaning $\frac{|0\rangle + |1\rangle}{\sqrt{2}}$.

The CNOT gate is a two-qubit gate. It allows entanglement. The CNOT gate operates over two qubits, one of which is target qubit and one control qubit. If the control qubit is in state $|1\rangle$, the target qubit gets inverted (the Pauli X-gate is applied onto the target qubit). If the control qubit is in state $|0\rangle$, no changes appear. Because CNOT operates over two qubits, the space dimension is 4, and it can not be represented on the Bloch sphere. The CNOT matrix is presented in Equation 1.

$$\mathrm{CNOT} = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\end{pmatrix}$$

Equation 1. The CNOT matrix

The Oracle function is an abstract black box operation within a quantum algorithm. It is a unitary operation U_f based on the principles of the CNOT gate. It is defined as presented in Equation 2.

$$\mathrm{U_f}\,|\,\mathrm{x}\rangle\,|\,\mathrm{y}\rangle = |\,\mathrm{x}\rangle\,|\,\mathrm{y} \oplus \mathrm{f(x)}\rangle$$

Equation 2. The Oracle function

where x is the control qubit, and y is the target qubit, and $\oplus$ marks the XOR operation. The key mechanism of Oracle function is phase kickback, where Oracle codes the result of the function in the control qubit's phase, without changing its values directly. In that way, after some quantum transformations, the phase is transformed into measurable probability, e.g., the global properties of the function. The main idea within Oracle function is the superposition of the input, which allows quantum acceleration, meaning a quantum computer works with all inputs at the same time.

Measurement in quantum computing is a special process because it collapses the quantum state and transforms the superposition into a classical result. If a qubit is in state $|\psi\rangle = a|0\rangle + b|1\rangle$, the measurement returns 0 with probability $|a|^2$ or 1 with probability $|b|^2$ after which the state irreversibly collapses to the corresponding basic state. The measurement is not a reversible operation, like other quantum gates, and therefore is performed at the end of a quantum algorithm. The main difference between classical and quantum computing is that classical measurement reveals an existing bit value, while quantum measurement physically changes the state of the system.

The graphical representation of the quantum operations, as they appear in quantum algorithms are given in Figure 2.

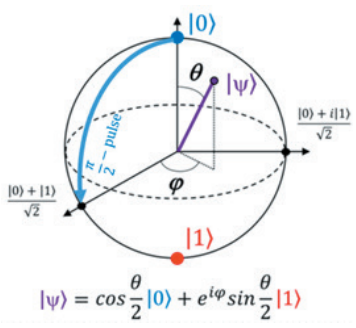

Figure 1. The Bloch sphere

Figure 6. Printed sample label with QR code

3. QUANTUM ALGORITHMS

Quantum algorithms represent a series of quantum gates on qubits, and the measurement in the end, where the probability of the result can be increased by interference. The calculations are based on superposition, interference, and entanglement, applied to state vectors within quantum space. Solving problems in this way, using quantum operations, is significantly faster, and in some cases, almost impossible to perform on classic computers. The core of quantum computing is the ability of quantum algorithms to use the principles of quantum mechanics, like interference and parallel processing of information. Some of the most well-known quantum algorithms are Deutsch algorithm (determine if the function is constant or balanced for only one evaluation of the function, classical computers need two evaluations), Grover algorithm (for searching large databases), Shor algorithm (factorisation of large numbers, compromises classical cryptography), VQE and QAOA (simulation of quantum systems and chemical molecules, classical computers can hardly solve this kind of problems because of exponential states growth).

3.1. DEUTSCH ALGORITHM

Problem: *The function f:{0,1}→{0,1} is given. Determine if the function is constant (output 0) or balanced (output 1).*

The Deutsch algorithm solves the Deutsch problem, which examines if the function is constant ($f(0)=f(1)$) or balanced ($f(0)\neq f(1)$), [4]. The classical approach needs two evaluations of functions $f(0)$ and $f(1)$. Quantum approach using the Deutsch algorithm needs only one evaluation.

First, the algorithm starts with two qubits, one target and one control qubit. On both qubits, the Hadamard gate is applied to create a superposition, Equation 3.

$$|\pi_1\rangle = |+\rangle\,|-\rangle = \frac{1}{2}(|0\rangle - |1\rangle)\,|0\rangle + \frac{1}{2}(|0\rangle - |1\rangle)\,|1\rangle$$

Equation 3. Application of Hadamard gate

Then, the Oracle function is applied to enable quantum parallelism, Equation 3.

$$|\pi_2\rangle = \frac{1}{2}(|0\oplus f(0)\rangle - |1\oplus f(0)\rangle)\,|0\rangle + \frac{1}{2}(|0\oplus f(1)\rangle - |1\oplus f(1)\rangle)\,|1\rangle$$

Equation 3. Application of the Oracle function

Now, observe the following expressions, Equation 4:

$$(1)\quad |0\oplus a\rangle - |1\oplus a\rangle = (-1)^a(|0\rangle - |1\rangle)$$

$$(2)\quad |0\oplus 0\rangle - |1\oplus 0\rangle = |0\rangle - |1\rangle = (-1)^0(|0\rangle - |1\rangle)$$

$$(3)\quad |0\oplus 1\rangle - |1\oplus 1\rangle = |1\rangle - |0\rangle = (-1)^1(|0\rangle - |1\rangle)$$

Equation 4. Explanation of XOR application

If (1), (2), and (3) are applied to $|\pi_2\rangle$, the following conclusions are obtained, Equation 5:

$$|\pi_2\rangle = \frac{1}{2}(-1)^{f(0)}(|0\rangle - |1\rangle)\,|0\rangle + \frac{1}{2}(-1)^{f(1)}(|0\rangle - |1\rangle)\,|1\rangle = |-\rangle\left(\frac{(-1)^{f(0)}\,|0\rangle + (-1)^{f(1)}\,|1\rangle}{\sqrt{2}}\right) \quad (4)$$

$$|\pi_2\rangle = (-1)^{f(0)}\,|-\rangle\left(\frac{|0\rangle + (-1)^{f(0)\oplus f(1)}\,|1\rangle}{\sqrt{2}}\right) = \begin{cases} (-1)^{f(0)}\,|-\rangle\,|+\rangle & \text{if } f(0)\oplus f(1) = 0 \\ (-1)^{f(1)}\,|-\rangle\,|-\rangle & \text{if } f(0)\oplus f(1) = 1 \end{cases} \quad (5)$$

Equation 5. Application of XOR

After the Oracle, the Hadamard gate is applied on the first qubit in order to create interference, Equation 6.

$$|\pi_3\rangle = \begin{cases} (-1)^{f(0)}\,|-\rangle\,|0\rangle & \text{if } f(0)\oplus f(1) = 0 \\ (-1)^{f(0)}\,|-\rangle\,|1\rangle & \text{if } f(0)\oplus f(1) = 1 \end{cases}$$

Equation 6. Application of Hadamard gate on the first qubit

Then the measurement is applied to the first qubit. If the result is 0, the function is constant. If the result is 1, the function is balanced, Figure 3.

4. EXPERIMENT

The quantum algorithm can be executed on different platforms. Since installing and maintaining the necessary software on a local computer is relatively complicated, we decided to use Google Colab [5]. Google Colab is a hosted service of Jupyter Notebook [6].

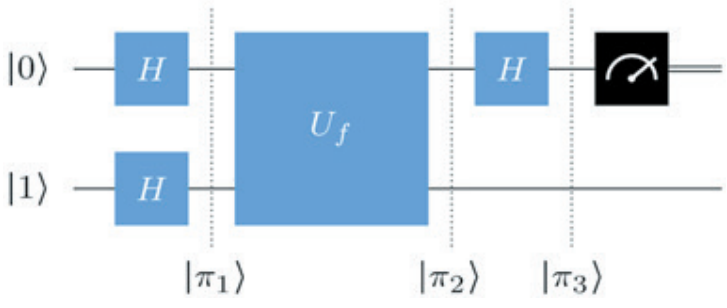

Figure 3. Deutsch algorithm

No setup is required to work in it. It provides free access to the necessary computing resources, including Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs). Google Colab requires a regular Gmail account.

To execute our quantum Deutsch's algorithm, we use Qiskit [7], and it is a Python SDK. Qiskit is an open-source software and development kit that enables working with quantum computers at the level of circuits, pulses, and algorithms. It provides tools for developing and running quantum programs, and is supported on Google Colab. Also, in cooperation with other libraries, it provides useful possibilities for working with graphics and simulators. To execute our quantum algorithm, it is necessary to install Qiskit and additional libraries, which is achieved by executing the program code in Colab given in Listing 1.

In the first cell, Colab executes the following program that forms a quantum circuit, performs a measurement, and plots the quantum circuit. It is necessary to uncomment either the Constant function (A) or the balanced function (B). The above code can be seen in Listing 2. The drawn quantum circuits are shown in Figure 4.

In order to show the results of the measurements in both of these cases, we decided to display them using a print command. The simulation and result printing are achieved by executing the program code given in Listing 3. The result printings for both of these cases are shown in Listing 3 and Listing 4.

As expected, the results correctly showed 0 for constant and 1 for balanced function in 100% of cases. This is possible because it is a simulator of the work of a quantum computer, which does not show the calculation errors that exist in a real quantum computer.

```
!pip install qiskit
!pip install qiskit_aer
!pip install pylatexenc
!pip install matplotlib-venn
```

Listing 1. Commands to install Qiskit and additional libraries

```
from qiskit import QuantumCircuit
from qiskit import transpile
from qiskit_aer import AerSimulator
# Preparation: Making a quantum circuit with 2 qubits (q0) and (q1) and a classical register c
qc = QuantumCircuit(2,1)
# Preparation: ancilla (q1) on |1>, Hadamard gate on both (q0) and (q1)
qc.x(1)
qc.h(0)
qc.h(1)
qc.barrier() # First barrier
# Orakl direktno u funkciji
# qc.x(1)     # A) Example of a constant function - uncomment to make it an Oracle function
# qc.cx(0,1)  # B) Example of a balanced function - uncomment to make it an Oracle function
qc.barrier() # Second barrier
# Hadamard gate on input qubit (q0) and measurement
qc.h(0)
qc.measure(0,0) # The result of the measurement goes into the classic register c
qc.draw('mpl') # Drawing a quantum circuit
```

Listing 2. Program that forms a quantum circuit, performs a measurement, and plots the quantum circuit

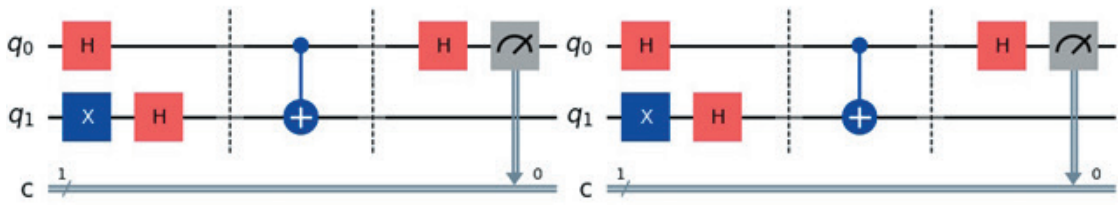

Figure 4. Drawn quantum circuits with constant (left) and balanced (right) function

```
simulator = AerSimulator()
compiled_circuit = transpile(qc, simulator)
shoots = 1024
sim_result = simulator.run(compiled_circuit, shoots = shoots).result()
counts = sim_result.get_counts()
print(counts)
```

Listing 3. Simulation and histogram plotting

```
{'0': 1024}
```

Listing 4. The result obtained for a constant function

```
{'1': 1024}
```

Listing 5. The result obtained for a balanced function

5. CONCLUSION

Our paper explains the basic principles of quantum computing. This includes the fields of quantum mechanics, mathematics, and computer science. The phenomena of superposition, interference, and entanglement are explained. The basics of this field, including the qubit, quantum gates, and quantum operations, are explained. The quantum Deutsch algorithm is presented and explained in mathematical form before it is executed. Prerequisites for the execution of the quantum Deutsch's algorithm are presented. Python code listings are provided with comments. Images of quantum circuits and execution results are also given, from which it can be seen that the algorithm was successfully executed. In the future, there is a possibility of working with other algorithms as well as executing algorithms on a real quantum computer.

REFERENCES

[1] M. Horowitz and E. Grumbling, "Quantum computing: progress and prospects," 2019.

[2] S. Prashant, "Study on the basics of Quantum Computing," 2007.

[3] T. G. Wong, "Introduction to classical and quantum computing," 2022.

[4] "Deutsch's algorithm," IBM, [Online]. Available: https://quantum.cloud.ibm.com/learning/en/courses/fundamentals-of-quantum-algorithms/quantum-query-algorithms/deutsch-algorithm. [Accessed 15 4 2026].

[5] "Welcome To Colab," [Online]. Available: https://colab.research.google.com/. [Accessed 15 4 2026].

[6] "Jupyter Notebook," [Online]. Available: https://jupyter.org/. [Accessed 15 4 2026].

[7] "Introduction to Qiskit," [Online]. Available: https://quantum.cloud.ibm.com/docs/en/guides/tools-intro. [Accessed 15 4 2026].