The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM)
ISSN: 2602-3199

The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM), 2023

Volume 26, Pages 624-632

IConTES 2023: International Conference on Technology, Engineering and Science

Roadmap for Simulating Quantum Circuits Utilising IBM's Qiskit Library: Programming Approach

Yousef Jaradat
Al-Zaytoonah University of Jordan

Mohammad Alia
Al-Zaytoonah University of Jordan

Mohammad Masoud
Al-Zaytoonah University of Jordan

Ahmad Mansrah
Al-Zaytoonah University of Jordan

Ismael Jannoud
Al-Zaytoonah University of Jordan

Omar Alheyasat
AlBalqa' Applied University

Abstract: This paper explains the roadmap for running quantum circuit programs based on the Qiskit library in quantum simulators as well as real cloud-based IBM quantum computers. Qiskit is a free and open-source software development platform based on the Python programming language that is used in quantum programming. Qiskit acts as a link between quantum computing's theoretical foundations and the practical aspects of programming and experimentation. It also allows users to experiment with and develop quantum algorithms, as well as simulate and execute them on simulators and real-world cloud-based quantum devices. It also simplifies the quantum programming process and allows a diverse range of people to participate in the exciting world of quantum computing. The paper, on the other hand, provides the mathematical foundation for analyzing quantum circuits and algorithms using linear algebra principles, as they provide the tools needed to describe and manipulate quantum states and operations. Furthermore, the paper shows quantum circuit design and implementation using real Qiskit codes.

Keywords: Qiskit, Quantum circuit, Quantum algorithm, Entanglement, IBM

Introduction

Quantum computing is a type of computer that processes information using quantum mechanics principles. Quantum computers are substantially quicker than conventional computers at solving specific sorts of problems, such as breaking encryption, simulating physical systems, and discovering novel pharmaceuticals (Gill et al., 2023). Information is processed in classical computing using bits that can represent either as 0 or a 1. Quantum computing, on the other hand, employs quantum bits, or qubits, which can exist in a state of superposition, expressing both 0 and 1 at the same time (Preskill, 2021; Hidary & Hidary, 2019). Some of the key ideas in quantum computing (Nielsen & Chuang, 2010; Gyongyosi & Imre, 2019):

- Superposition: Unlike classical bits, which can only exist in one of two states (0 or 1) at any time, qubits can exist in a superposition of both states at the same time. This enables quantum computers to do several calculations at the same time, potentially speeding up certain algorithms.
- Entanglement occurs when the state of one qubit becomes intertwined with the state of another, regardless of their physical distance. This phenomenon allows quantum computers to process information in a highly correlated and interconnected manner, which can be used to perform specialized computations.

The most prevalent types of quantum computers being developed are gate-based quantum computers and adiabatic quantum computers (Bruzewicz et al., 2019; Albash & Lidar, 2018). Gate-based quantum computers operate qubits using a series of quantum gates, whereas adiabatic quantum computers find the ground state of a quantum system via a process known as adiabatic evolution. Quantum computing is still in its early stages of research and development, but it has the potential to transform many sectors. The following are some of the potential uses of quantum computing (Hassija et al., 2020):

- Breaking encryption: Quantum computers have the potential to break the encryption employed in many modern security systems, such as credit card and online banking systems.
- Physical system simulation: Quantum computers could be used to simulate the behavior of physical systems like molecules and materials. This might be used to develop novel treatments and materials, as well as to investigate the effects of climate change.
- Finding new pharmaceuticals: By simulating the interactions of molecules, quantum computers could be utilized to find novel drugs. This might significantly speed up the drug discovery process.
- Optimization: Quantum computers could be used to address optimization problems like finding the shortest path between two points or determining the best method to allocate resources. This could be applied to transportation, logistics, and manufacturing.

Qiskit is a free and open-source software development (SDK) platform for programming, simulation, or and performing quantum computations on real-world quantum hardware (Wille et al., 2019; Fingerhuth et al., 2018). It is one of the most prominent frameworks for working with quantum computers, invented by IBM. Qiskit is a suite of tools and components (Python libraries) that allows researchers, developers, and programmers to work with quantum computing algorithms and experiments. Qiskit is made up broadly of four core modules:

- *Qiskit Terra*: is the foundation of the Qiskit framework. It serves as the foundation for quantum circuits, gates, and operators. Users can create quantum circuits with numerous gate types and apply operations on qubits. Terra also provides circuit visualization and optimization tools.
- *Qiskit Aer*: This is a quantum circuit simulator. It can be used to test and debug quantum programs before they are executed on actual hardware. Aer also offers tools for modelling noise's impact on quantum circuits.
- *Qiskit Ignis*: This is a set of error-mitigation tools. It can be used to increase the accuracy of quantum programs by compensating for noise errors.
- *Qiskit Aqua*: is a quantum algorithm development framework. It contains libraries for many different quantum algorithms, as well as tools for optimizing and benchmarking these algorithms.

Qiskit is an elegant tool for building and testing quantum computing algorithms. Researchers and developers from all over the world use it to investigate the potential of quantum computing for a wide range of applications. The rest of the paper is organized as follows. Section II provides an overview of the quantum circuits. Quantum circuits programming details for both simulators and real quantum computers are provided in section III. Section IV concludes the paper.

Quantum Circuits

Basic Concepts

A quantum circuit is a set of quantum gates that are applied to a group of qubits (Cervera-Lierta et al., 2019; Fisher et al., 2023). In quantum computing, qubit is the fundamental computational unit. Quantum circuits are used to create quantum algorithms that can only be run on a quantum computer. Depending on the quantum algorithm used, quantum gates in any quantum circuit can be connected in series or parallel or both. Quantum circuits have the following basic components:

- Qubits: In quantum computing, qubits are the fundamental unit of information. They can be in a state that is a superposition of two states, 0 and 1.
- Quantum gates are the fundamental operations that can be done on qubits. They can be employed to spin a qubit's state or to entangle two qubits.
- Measurements are used to determine the state of a qubit. When a qubit is measured, it collapses into one of two states: 0 or 1.

Quantum circuits are depicted graphically, with qubits denoted by lines and quantum gates denoted by boxes. The sequence of the gates in the circuit is critical because it dictates how the gates are applied to the qubits. Fig. 1 shows a sample of quantum circuits that is used throughout this paper. This circuit is produced using the IBM's quantum composer platform.

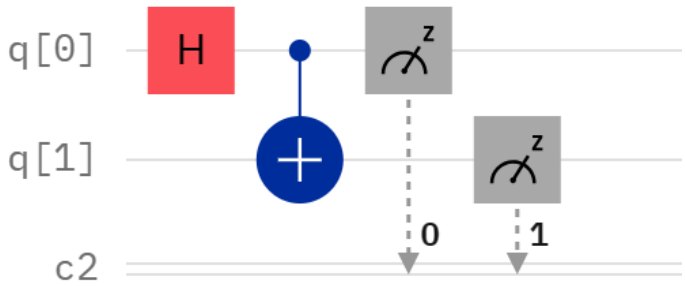

Figure 1. Entangled quantum circuit using IBM's composer platform

Quantum circuits can be used to construct a wide range of quantum algorithms, including Shor's algorithm for integer factoring, Grover's algorithm for searching an unsorted database, and quantum simulation of physical systems. Quantum circuits have the following essential properties:

- Quantum circuits are acyclic: This signifies that the circuit has no loops. A loop is a gate sequence that begins and ends at the same qubit. Loops are not permitted in quantum circuits because they can induce computing errors.
- Fan-in and fan-out are not permitted in quantum circuits. The number of qubits that can be acted on by a single gate is known as fan-in, and the number of qubits that can be influenced by a single gate is known as fan-out. Fan-in and fan-out in quantum circuits are normally limited to 1.
- Quantum circuits are reversible: Quantum circuit reversibility is a fundamental characteristic of quantum physics. It is critical for the development of quantum computing and offers a wide range of possible applications. Quantum reversibility requires that the number of inputs for any quantum circuits is equal to the number of outputs. The reason for quantum circuit reversibility is that the wavefunction that represents the state of he quantum system obeys the Schrodinger equation, which is a reversible equation. This means that it is possible to go from any state of the system to any other state. In addition, quantum gate is a unitary operator, which means it retains the wavefunction's norm. This means that the quantum gate is reversible.
- quantum measurement is irreversible: This is because measuring a qubit causes its wavefunction to collapse into a fixed state. This means that the knowledge about the qubit's other potential states is lost.

Mathematical Analysis

Qubits can be represented using vectors in Hilbert space. Dirac notation is used to represents qubits [3]. Unitary matrices can be used to study quantum circuits (Cruz-Lemus et al., 2021). A unitary matrix is one that maintains the inner product of vectors. This means that multiplying a vector by a unitary matrix has no effect on its length. The unitary matrix for a quantum circuit is the product of the unitary matrices for the individual gates in the circuit. The direction of the product is from right to left. Tensor product is utilized to multiply two or more parallel gates. The final state ($|\psi_f\rangle$) of the circuit shown in Figure 1 is given by:

$$|\psi_f\rangle = CNOT(H \otimes I)(q[0] \otimes q[1]) \quad (1)$$

if $q[0]$ and $q[1]$ are both initialized to $|0\rangle$, then the equation will be:

$$|\psi_f\rangle = CNOT(H \otimes I)|00\rangle \quad (2)$$

CNOT, Hadamard, and identity gates are given by the following unitary matrices.

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \; H = 1/\sqrt{(2)} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \; I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Then, equation 2 is given by:

$$|\psi_f\rangle = 1/2 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

after reduction, the equation above is given by:

$$|\psi_f\rangle = 1/2 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = 1/2(|00\rangle + |11\rangle)$$

which is a perfect entangled state, usually referred to as a Bells state. Bell states are an essential component of quantum ´ information science. Many quantum algorithms, such as quantum teleportation and quantum cryptography, make use of them.

Qiskit Programming Roadmap

This section describes how to run Qiskit-based quantum circuit programs on both a quantum simulator and IBM's real cloud-based quantum computer.

1) To begin, run the following command in a *Jupyter* notebook terminal to quickly install the library. This will install the most recent stable version of the library. It is recommended to install Qiskit with the extra visualization if you intend to use visualization functionality or *Jupyter* notebooks. The Qiskit library was install on August 20th, 2023.

```
pip install qiskit[visualization]
```

2) To check the version of Qiskit and its components in the Notebook by running the following command:

```
import qiskit
qiskit.__qiskit_version__
```

3) To implement the quantum circuit shown in Figure 1, execute the following code:

```
from qiskit import
QuantumCircuit
qc = QuantumCircuit(2,2)
qc.barrier()
qc.h(0)
qc.barrier()
qc.cx(0,1)
qc.measure([0,1],[0,1])
qc.draw(output='mpl')
```

The code utilizes the *matplotlib* library to produce much better circuit visualization as shown in Figure 2.

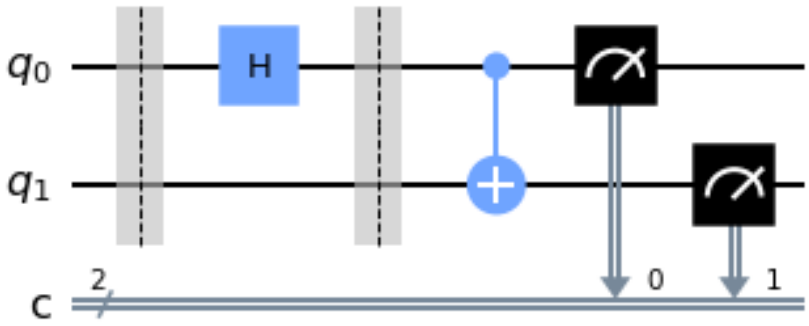

Figure 2. Entangled quantum circuit using Qiskit library

4) To confirm the entanglement of the two qubits, the code must be performed on either a quantum simulator or a real quantum computer platform. Quantum simulator can run quantum circuit programs on your local machine or on the cloud. Qiskit *Aer* is the simulator used to run and debug quantum programs before executing them on real quantum computers. First, let us run the given code on Aer simulator installed on a local machine. To get a list of the available simulators, the following statement should be executed.

```
from qiskit import Aer
Aer.backends()
```

The above statement should produce something similar to the following list, as illustrated in Figure 3.

```
[AerSimulator('aer_simulator'),
 AerSimulator('aer_simulator_statevector'),
 AerSimulator('aer_simulator_density_matrix'),
 AerSimulator('aer_simulator_stabilizer'),
 AerSimulator('aer_simulator_matrix_product_state'),
 AerSimulator('aer_simulator_extended_stabilizer'),
 AerSimulator('aer_simulator_unitary'),
 AerSimulator('aer_simulator_superop'),
 QasmSimulator('qasm_simulator'),
 StatevectorSimulator('statevector_simulator'),
 UnitarySimulator('unitary_simulator'),
 PulseSimulator('pulse_simulator')]
```

Figure 3. List of local Aer quantum simulators

Then we choose one of these simulators, *transpile* the quantum circuit to match the constraints and characteristics of the target quantum simulator/device and produce the results of the entangled quantum circuit as shown in the following code:

```
sim = Aer.get_backend('qasm_simulator')
from qiskit import transpile
qc_trans = transpile(qc, sim)
result = sim.run(qc_trans).result()
state_counts = result.get_counts(qc_trans)
from qiskit.visualization import plot_histogram
plot_histogram(state_counts,title='Bell-State counts')
```

Figure 4 shows the result of the above code, and it is compatible with equations above, with a probability of almost 0.5 for each of the entangled state.

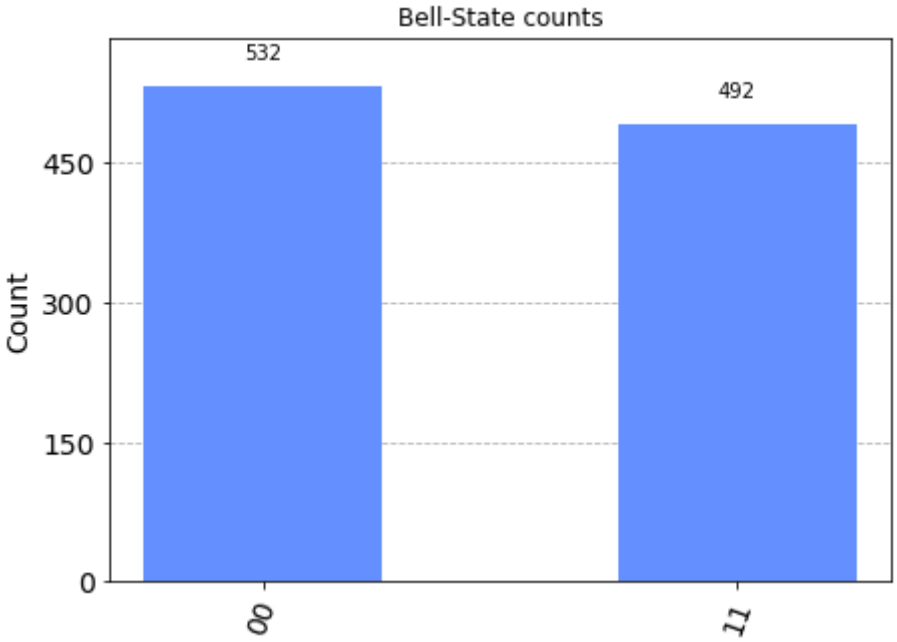

Figure 4. Histogram of entangled states (counts)

To get the probability of each quantum state, Qiskit introduced primitives which are the smallest processing instruction for a given abstraction level. *Sampler* is a Qiskit primitive that calculates probabilities or quasi-probabilities of quantum states. Quasi-probabilities are similar to regular probabilities except they may include negative values, which can occur when using certain error mitigation techniques. The code below shows how to calculate the quasi-probabilities of the Bell circuit. Figure 5 shows the corresponding probability distribution of the quantum states.

```
from qiskit_aer.primitives import Sampler
sampler = Sampler()
job = sampler.run(qc_trans)
plot_histogram(job.result().quasi_dists, title='Bell-State prob.')
```

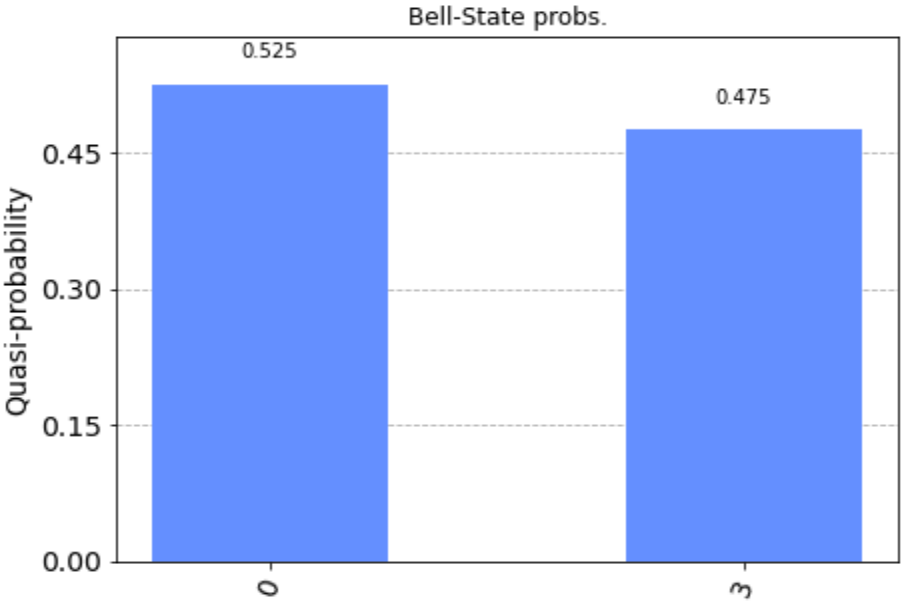

Figure 5. Histogram of entangled states probabilities

5) To run the quantum circuit code on IBM's online quantum computer systems, the following code should be run:

```
from qiskit_ibm_runtime import QiskitRuntimeService, Sampler
IBMTOKEN = "********" # IBM API Token
service = QiskitRuntimeService( channel='ibm_quantum', token=IBMTOKEN)
```

The code above will import the *QiskitRuntimeService* class and use it to connect the IBM cloud-based quantum computer through your IBM API token. The circuit code then can be run on a simulator or real quantum system. To get a list of real operational quantum computers, the following code is used.

```
service_real = QiskitRuntimeService( channel='ibm_quantum', token=tkn)
be_real = service_real.backends(simulator=False, operational=True)
```

Figure 6 shows a list of the operational real quantum systems.

```
[<IBMBackend('ibmq_quito')>,
 <IBMBackend('ibmq_lima')>,
 <IBMBackend('ibmq_belem')>,
 <IBMBackend('ibm_lagos')>,
 <IBMBackend('ibm_perth')>,
 <IBMBackend('ibmq_jakarta')>,
 <IBMBackend('ibmq_manila')>,
 <IBMBackend('ibm_nairobi')>]
```

Figure 6. Real quantum computers backends

To check the queue of pending jobs, number of qubits of the quantum computer backends, the following code is used.

```
for i in range(len(be_real)):
    be_name = be_real[i].name
    be_qb = be_real[i].num_qubits
    be_pj = be_real[i].status(). to_dict()['pending_jobs']
   print(f'{be_name :<15}: {be_pj :<5}, num_qubits: {be_qb}')
```

The output of the above code is shown in Figure 7.

```
ibmq_quito      :  542  , num_qubits:  5
ibmq_lima       :  364  , num_qubits:  5
ibmq_belem      :  305  , num_qubits:  5
ibm_lagos       :  148  , num_qubits:  7
ibm_perth       :  123  , num_qubits:  7
ibmq_jakarta    :  313  , num_qubits:  7
ibmq_manila     :  4111 , num_qubits:  5
ibm_nairobi     :  204  , num_qubits:  7
```

Figure 7. Pending jobs and number of qubits on the backends

To selected the least busy real quantum computer to run your quantum circuit, the following code is used.

```
backend_r = service_real.least_busy (simulator=False,operational=True)
print(backend_r.name)
```

In our case, the least busy computer to run the quantum circuit is the *ibmq perth* quantum computer. The quantum circuit program is then executed on the real quantum computer backend. Most of the time, the result is not as immediate as it is on the simulator backend. Normally, the program is queued before being executed. So, checking the program status is necessary as it may take a long time before it is executed. The following code provides the necessary steps for executing and monitoring the program.

```
sampler_r = Sampler(backend_r) job_r = sampler_r.run(qc, shots=4000)
job_r.status()
plot_histogram(job_r.result().quasi_dists,title='Bell-State prob.')
```

It is important to run the program many times as real quantum computers suffer from noise. In the code above the program was run 4000 times before getting the results as shown in Figure 8. Noise has its own effect on the entangled states. Negative low probabilities are noticed on the |01⟩ and |10⟩ quantum states.

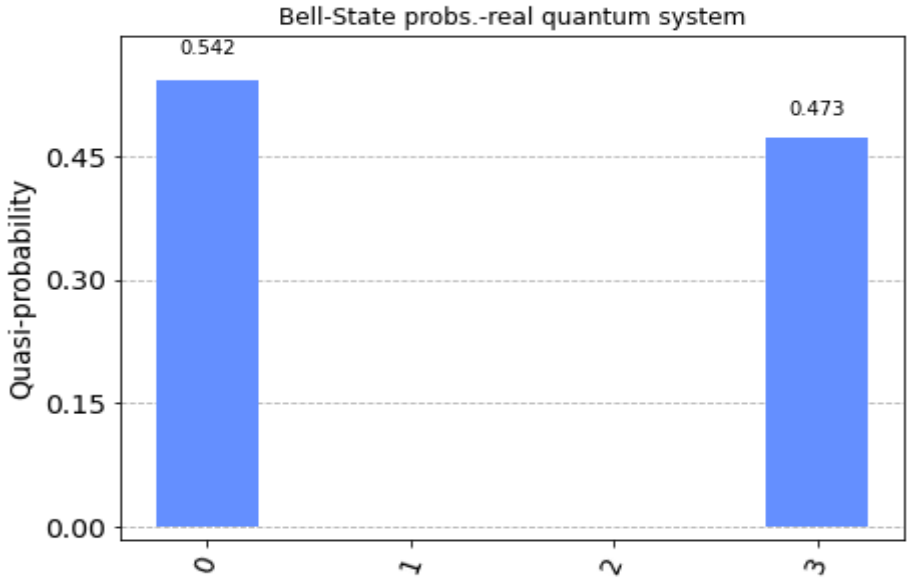

Figure 8. Quasi-probabilities of different quantum states

Conclusion

The paper includes programming details for building quantum applications using the IBM's Qiskit toolkit. It demonstrates the programming steps required to build quantum gates, quantum circuits, and quantum algorithms. It additionally illustrates how to simulate quantum circuits in quantum simulator backends. Moreover, it describes how to execute quantum circuits in a real cloud-based IBM quantum computer. Furthermore, it demonstrates how to choose the least busy real quantum computer to run the circuit program.

Scientific Ethics Declaration

The authors declare that the scientific ethical and legal responsibility of this article published in EPSTEM journal belongs to the authors.

Acknowledgements or Notes

* This article was presented as an oral presentation at the International Conference on Technology, Engineering and Science (www.icontes.net) held in Antalya/Turkey on November 16-19, 2023.

* The authors would like to thank the Deanship of scientific research and Innovation at Al-Zaytoonah University of Jordan (ZUJ) for Funding this work through ZUJ research fund No. (41/17/2022-2023)

References

Albash, T., & Lidar, D. A. (2018). Adiabatic quantum computation. *Reviews of Modern Physics*, *90*(1), 015002.

Bruzewicz, C. D., Chiaverini, J., McConnell, R., & Sage, J. M. (2019). Trapped-ion quantum computing: Progress and challenges. *Applied Physics Reviews*, *6*(2).

Cervera-Lierta, A., Latorre, J. I., & Goyeneche, D. (2019). Quantum circuits for maximally entangled states. *Physical Review A*, *100*(2), 022342.

Cruz-Lemus, J. A., Marcelo, L. A., & Piattini, M. (2021). Towards a set of metrics for quantum circuits understandability. In *International Conference on the Quality of Information and Communications Technology* (pp. 239-249). Cham: Springer International Publishing.

Fingerhuth, M., Babej, T., & Wittek, P. (2018). Open source software in quantum computing. *PloS one*, *13*(12), e0208561.

Fisher, M. P., Khemani, V., Nahum, A., & Vijay, S. (2023). Random quantum circuits. *Annual Review of Condensed Matter Physics*, *14*, 335-379.

Gill, S. S., Kumar, A., Singh, H., Singh, M., Kaur, K., Usman, M., & Buyya, R. (2022). Quantum computing: A taxonomy, systematic review and future directions. *Software: Practice and Experience*, *52*(1), 66-114.

Gyongyosi, L., & Imre, S. (2019). A survey on quantum computing technology. *Computer Science Review*, *31*, 51-71.

Hassija, V., Chamola, V., Saxena, V., Chanana, V., Parashari, P., Mumtaz, S., & Guizani, M. (2020). Present landscape of quantum computing. *IET Quantum Communication*, *1*(2), 42-48.

Hassija, V., Chamola, V., Goyal, A., Kanhere, S. S., & Guizani, N. (2020). Forthcoming applications of quantum computing: peeking into the future. *IET Quantum Communication*, *1*(2), 35-41.

Hidary, J. D., & Hidary, J. D. (2019). *Quantum computing: an applied approach* (Vol. 1). Cham: Springer.

Nielsen, M. A., & Chuang, I. L. (2010). *Quantum computation and quantum information*. Cambridge university press.

Preskill, J. (2021). Quantum computing 40 years later. In *Feynman lecturures on* computation (pp. 193-244). CRC Press.

Wille, R., Van Meter, R., & Naveh, Y. (2019). IBM's Qiskit tool chain: Working with and developing for real quantum computers. In 2019 *Design, Automation & Test in Europe Conference & Exhibition (DATE)* (pp. 1234-1240). IEEE.

Author Information

Yousef Jaradat
Al-Zaytoonah University of Jordan
Amman - Jordan
Contact e-mail: *y.jaradat@zuj.edu.jo*

Mohammad Alia
Al-Zaytoonah University of Jordan
Amman - Jordan

Mohammad Masoud
Al-Zaytoonah University of Jordan
Amman - Jordan

Ahmad Mansrah
Al-Zaytoonah University of Jordan
Amman - Jordan

Ismael Janooud
Al-Zaytoonah University of Jordan
Amman - Jordan

Omar Alheyasat
AlBalqa' Applied University
Salt-Jordan

To cite this article:

Jaradat, Y., Mohammad, A., Masoud, M., Mansrah, A., Jannoud, I., & Alheyasat, O. (2023). Roadmap for simulating quantum circuits utilising ibm's qiskit library: programming approach. *The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM), 26*, 624-632.