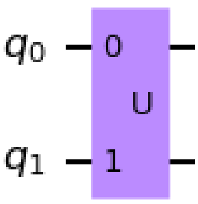
Have you ever thought about minding your own business?

On the “undesired” application of the “reset” instruction with Qiskit

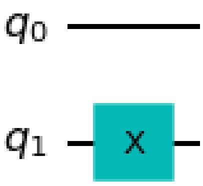
Claudio Fontana,
University of Palermo
Palermo, Italy
claudio.fontana@unipa.it

Let’s assume we want to build a unitary operator that performs the linear transformation from a set of zeros qubits to an arbitrary state vector.

E.g. n_qubits=2; SV=[0,0,1,0] (i.e. |10> w.r.t. the standard basis):



that, if decomposed, is

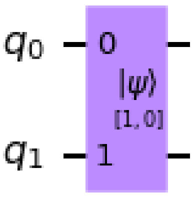


Our goal is to obtain the unitary operator U that realizes the linear transformation:

$$U|00\rangle = SV$$

The simplest way to initialize a circuit with Qiskit is to make use of the “initialize” method:

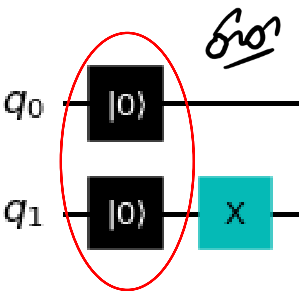
```
from qiskit import QuantumCircuit
circuit = QuantumCircuit(2)
#Initialize to |10>
circuit.initialize('10', circuit.qubits)
circuit.draw(output="mpl")
```



What’s the matter with “initialize”?

Let’s decompose the circuit:

```
circuit.decompose().draw(output="mpl")
```



 Claudio Fontana 16/09/2021

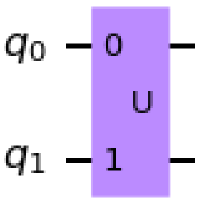
The problem with the “initialize” method is that it prepends a reset of the qubits that is a non-unitary operation, therefore it cannot give any matrix form:

```
from qiskit.opflow import CircuitOp
op=CircuitOp(circuit).to_matrix()

QiskitError: 'Cannot apply Instruction: reset' Use %tb to get the full traceback.
```

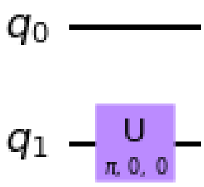
Solution:

Even if “deprecated”, at last, I found a solution: *a cool and refreshing glass of “aqua”.*

```
import qiskit.aqua as qa
from qiskit import QuantumCircuit
import numpy as np

n_qubits=2
circuit = QuantumCircuit(n_qubits)
SV=[0,0,1,0]
custom = qa.components.initial_states.Custom(num_qubits=n_qubits, state_vector=SV/
np.linalg.norm(SV))
custom_qubit_circuit = custom.construct_circuit()
custom_qubit_circuit.name='U'
circuit.append(custom_qubit_circuit, circuit.qubits)
print("Circuit")
circuit.draw(output="mpl")
```



that, if decomposed, is



In this way, we can obtain the desired matrix as:

```
from qiskit.visualization import array_to_latex
from qiskit.opflow import CircuitOp

op=CircuitOp(circuit).to_matrix()
array_to_latex(op, prefix="\\text{Operator = }\n")
```

$$\text{Operator} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

```
array_to_latex((op @ np.array([[1,0,0,0]]).T), prefix="\\text{SV = }\n")
```

$$\text{SV} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Q.E.D.

As an example, let's make the product between the operator end its adjoint (try to make the adjoint of an initialized circuit with "initialize" and you'll see what happens: it is impossible!!!):

```
op_adj=CircuitOp(circuit).adjoint().to_matrix()
array_to_latex(op_adj, prefix="\\text{Op_adjoint = }\n")
```

$$\text{Op\_adjoint} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}$$

```
array_to_latex((op @ op_adj), prefix="\\text{Op*Op_adjoint = }\n")
```

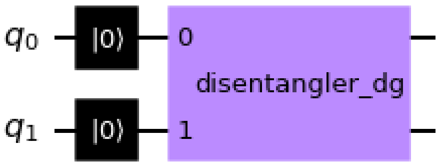
$$\text{Op*Op\_adjoint} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Q.E.D.

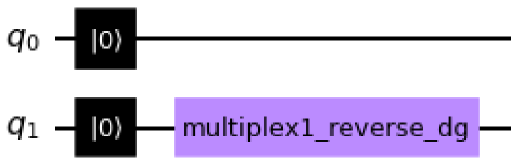
Non-deprecated solution #1:

A non-deprecated solution makes use of a RemoveResetInZeroState() method:

```
from qiskit.transpiler.passes import RemoveResetInZeroState
from qiskit import QuantumCircuit

n_qubits=2
circuit = QuantumCircuit(n_qubits)
SV=[0,0,1,0]
circuit.initialize(SV,circuit.qubits)
print("Circuit")
circuit.decompose().draw(output="mpl")
```

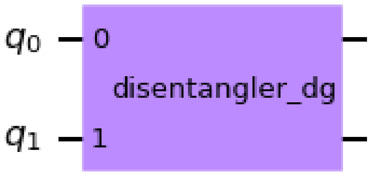


that, if decomposed, is:

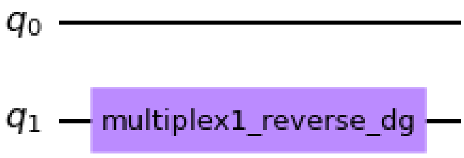


We can remove reset in zero state in a new circuit as follow:

```
tpass = RemoveResetInZeroState()
print("Circuit")
circ=tpass(circuit.decompose())
circ.draw(output="mpl")
```



that, if decomposed, is:



Now, we can obtain the desired matrix as:

```
from qiskit.visualization import array_to_latex
from qiskit.opflow import CircuitOp

op=CircuitOp(circ).to_matrix()
array_to_latex(op, prefix="\\text{circ = }\n")
```

$$\text{circ} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

```
array_to_latex((op @ np.array([[1,0,0,0]]).T), prefix="\\text{SV = }\n")
```

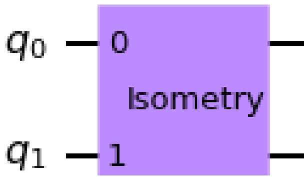
$$\text{SV} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Q.E.D.

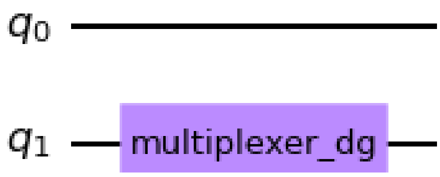
Non-deprecated solution #2:

A non-deprecated solution makes use of the isometry() method:

```
#see:https://qiskit-community.github.io/qiskit-translations/ml_IN/stubs/qiskit.circuit.QuantumCircuit.isometry.html
from qiskit import QuantumCircuit

n_qubits=2
circuit = QuantumCircuit(n_qubits)
SV=[0,0,1,0]
circuit.isometry(SV,list(range(n_qubits)),None)
print("Circuit")
circuit.draw(output="mpl")
```



that, if decomposed two times, is:



Now, we can obtain the desired matrix as:

```
from qiskit.visualization import array_to_latex
from qiskit.opflow import CircuitOp

op=CircuitOp(circuit).to_matrix()
array_to_latex(op, prefix="\\text{circuit = }\n")
```

$$\text{circuit} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

```
array_to_latex((op @ np.array([[1,0,0,0]]).T), prefix="\\text{SV = }\n")
```

$$\text{SV} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Q.E.D.

See also:

It is not possible to invert an initialized circuit · Issue #5976 · Qiskit/qiskit-terra · GitHubVar;

qiskit - What is the effect of the reset gate on the matrix form of a gate/circuit? - Quantum Computing Stack Exchange)