In [1]:

```
# Built-in modules
import math

# Imports from Qiskit
from qiskit import QuantumCircuit
from qiskit.circuit.library import GroverOperator, MCMT, ZGate
from qiskit.visualization import plot_distribution

# Imports from Qiskit Runtime
from qiskit_ibm_runtime import QiskitRuntimeService
from qiskit_ibm_runtime import SamplerV2 as Sampler
```

In [2]:
```
# To run on hardware, select the backend with the fewest number o
service = QiskitRuntimeService(channel="ibm_quantum")
backend = service.least_busy(operational=True, simulator=False)
backend.name
```

```
---------------------------------------------------------------------
--------
AccountNotFoundError                      Traceback (most recent ca
ll last)
Cell In[2], line 2
      1 # To run on hardware, select the backend with the fewest nu
mber of jobs in the queue
----> 2 service = QiskitRuntimeService(channel="ibm_quantum")
      3 backend = service.least_busy(operational=True, simulator=Fa
lse)
      4 backend.name

File /path/to/virtual/environment/lib/python3.12/site-packages/qisk
it_ibm_runtime/qiskit_runtime_service.py:124, in QiskitRuntimeServi
ce.__init__(self, channel, token, url, filename, name, instance, pr
oxies, verify, channel_strategy)
     81 """QiskitRuntimeService constructor
     82
     83 An account is selected in the following order:
   (...)
    120     IBMInputValueError: If an input is invalid.
    121 """
    122 super().__init__()
--> 124 self._account = self._discover_account(
    125     token=token,
    126     url=url,
    127     instance=instance,
    128     channel=channel,
    129     filename=filename,
    130     name=name,
    131     proxies=ProxyConfiguration(**proxies) if proxies else N
one,
    132     verify=verify,
    133     channel_strategy=channel_strategy,
    134 )
    136 self._client_params = ClientParameters(
    137     channel=self._account.channel,
    138     token=self._account.token,
   (...)
    142     verify=self._account.verify,
    143 )
    145 self._channel_strategy = channel_strategy or self._accoun
t.channel_strategy

File /path/to/virtual/environment/lib/python3.12/site-packages/qisk
it_ibm_runtime/qiskit_runtime_service.py:239, in QiskitRuntimeServi
ce._discover_account(self, token, url, instance, channel, filename,
name, proxies, verify, channel_strategy)
    237         if url:
    238             logger.warning("Loading default %s account. Inp
ut 'url' is ignored.", channel)
--> 239         account = AccountManager.get(filename=filename, nam
e=name, channel=channel)
    240 elif any([token, url]):
    241     # Let's not infer based on these attributes as they may
change in the future.
    242     raise ValueError(
    243         "'channel' is required if 'token', or 'url' is spec
ified but 'name' is not."
    244     )
```

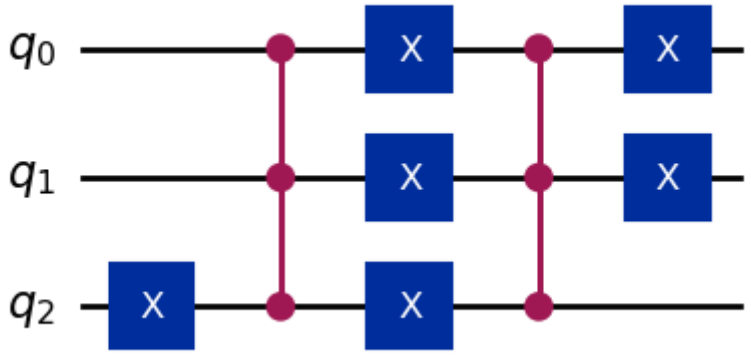
```
File /path/to/virtual/environment/lib/python3.12/site-packages/qisk
it_ibm_runtime/accounts/management.py:195, in AccountManager.get(cl
s, filename, name, channel)
    192     if account_name in all_config:
    193         return Account.from_saved_format(all_config[account
_name])
--> 195 raise AccountNotFoundError("Unable to find account.")

AccountNotFoundError: 'Unable to find account.'
```

In [3]:

```python
def grover_oracle(marked_states):
    """Build a Grover oracle for multiple marked states

    Here we assume all input marked states have the same number o

    Parameters:
        marked_states (str or list): Marked states of oracle

    Returns:
        QuantumCircuit: Quantum circuit representing Grover oracl
    """
    if not isinstance(marked_states, list):
        marked_states = [marked_states]
    # Compute the number of qubits in circuit
    num_qubits = len(marked_states[0])

    qc = QuantumCircuit(num_qubits)
    # Mark each target state in the input list
    for target in marked_states:
        # Flip target bit-string to match Qiskit bit-ordering
        rev_target = target[::-1]
        # Find the indices of all the '0' elements in bit-string
        zero_inds = [ind for ind in range(num_qubits) if rev_targ
        # Add a multi-controlled Z-gate with pre- and post-applie
        # where the target bit-string has a '0' entry
        qc.x(zero_inds)
        qc.compose(MCMT(ZGate(), num_qubits - 1, 1), inplace=True
        qc.x(zero_inds)
    return qc
```

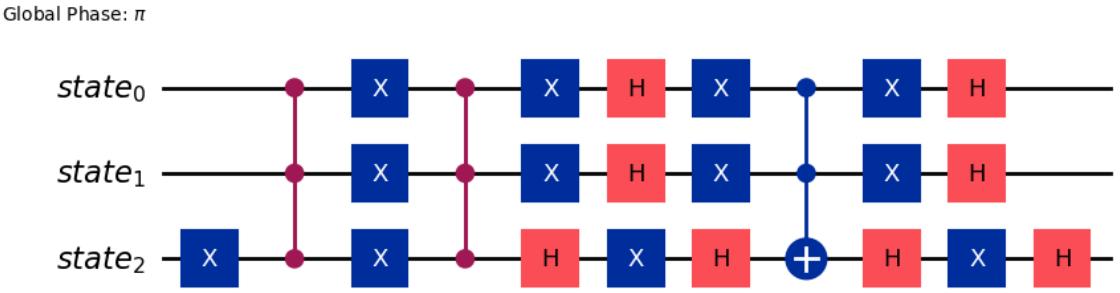
In [4]:
```
1 marked_states = ["011", "100"]
2
3 oracle = grover_oracle(marked_states)
4 oracle.draw(output="mpl", style="iqp")
```

Out[4]:



In [5]:
```
1 grover_op = GroverOperator(oracle)
2 grover_op.decompose().draw(output="mpl", style="iqp")
```

Out[5]:



In [6]:
```
1 optimal_num_iterations = math.floor(
2     math.pi / (4 * math.asin(math.sqrt(len(marked_states) / 2**g
3 )
```

In [ ]:
```
1
```