*ADITHYA DUTT KAMBHAMPATI*

*RA2011003011072*

**QUANTUM COMPUTATION CT 3 LAB ASSIGNMENT**

1. **Demonstrate Deutsch Algorithm for oracular function .**

Code:

import numpy as np

# Importing standard Qiskit libraries

from qiskit import QuantumCircuit, transpile, Aer, IBMQ, QuantumRegister, ClassicalRegister, execute, BasicAer

from qiskit.tools.jupyter import \*

from qiskit.visualization import \*

from ibm\_quantum\_widgets import \*

from qiskit.providers.aer import QasmSimulator

# Loading your IBM Quantum account(s)

provider = IBMQ.load\_account()

backend = BasicAer.get\_backend('qasm\_simulator')

shots=1024

qreg1 = QuantumRegister(2) # The quantum register of the qubits, in this case 2 qubits

register1 = ClassicalRegister(1)

qc = QuantumCircuit(qreg1, register1)

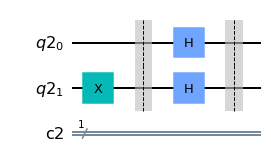
qc.x(1)

qc.barrier()

qc.h([0,1])

qc.barrier()

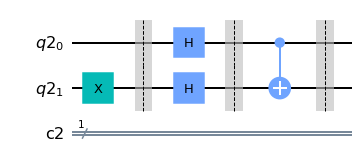
qc.draw()



qc.cx(0,1)

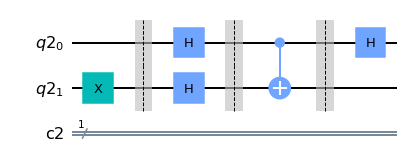
qc.barrier()

qc.draw()



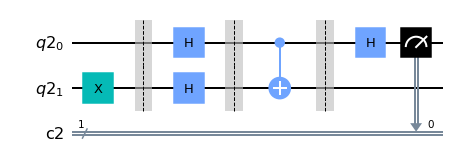
qc.h(0)

qc.draw()



qc.measure(qreg1[0],register1)

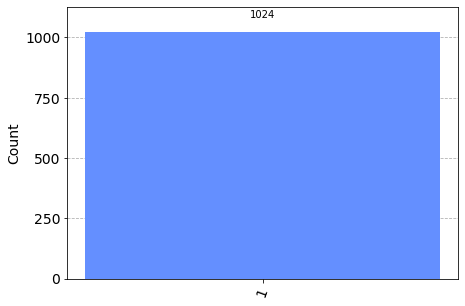
qc.draw()



results = execute(qc, backend=backend, shots=shots).result()

answer = results.get\_counts()

plot\_histogram(answer)



1. **Demonstrate Grover’s Algorithm for as the two-bit special string. Create the entire circuit and showcase how the circuit reveals the secret string hidden in the oracle.**

Code:

import numpy as np

# Importing standard Qiskit libraries

from qiskit import QuantumCircuit, transpile, Aer, IBMQ

from qiskit.tools.jupyter import \*

from qiskit.visualization import \*

from ibm\_quantum\_widgets import \*

from qiskit.providers.aer import QasmSimulator

# Loading your IBM Quantum account(s)

provider = IBMQ.load\_account()

#initialization

import matplotlib.pyplot as plt

import numpy as np

# importing Qiskit

from qiskit import IBMQ, Aer, assemble, transpile

from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister

from qiskit.providers.ibmq import least\_busy

# import basic plot tools

from qiskit.visualization import plot\_histogram

def initialize\_s(qc, qubits):

"""Apply a H-gate to 'qubits' in qc"""

for q in qubits:

qc.h(q)

return qc

n = 2

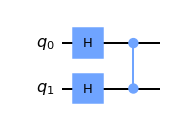
grover\_circuit = QuantumCircuit(n)

grover\_circuit = initialize\_s(grover\_circuit, [0,1])

grover\_circuit.draw()

grover\_circuit.cz(0,1) # Oracle

grover\_circuit.draw()



# Diffusion operator (U\_s)

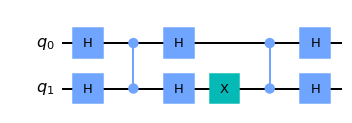
grover\_circuit.h([0,1])

grover\_circuit.x([1])

grover\_circuit.cz(1,0)

grover\_circuit.h([0,1])

grover\_circuit.draw()



sim = Aer.get\_backend('aer\_simulator')

# we need to make a copy of the circuit with the 'save\_statevector'

# instruction to run on the Aer simulator

grover\_circuit\_sim = grover\_circuit.copy()

grover\_circuit\_sim.save\_statevector()

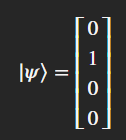
qobj = assemble(grover\_circuit\_sim)

result = sim.run(qobj).result()

statevec = result.get\_statevector()

from qiskit\_textbook.tools import vector2latex

vector2latex(statevec, pretext="|\\psi\\rangle =")



grover\_circuit.measure\_all()

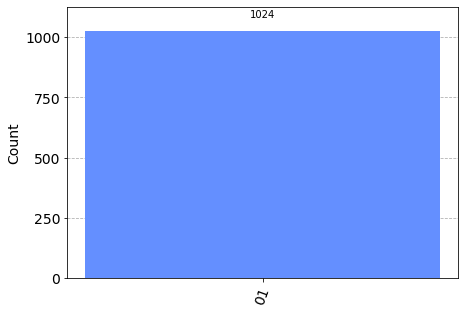
aer\_sim = Aer.get\_backend('aer\_simulator')

qobj = assemble(grover\_circuit)

result = aer\_sim.run(qobj).result()

counts = result.get\_counts()

plot\_histogram(counts)



# Load IBM Q account and get the least busy backend device

provider = IBMQ.load\_account()

provider = IBMQ.get\_provider("ibm-q")

device = least\_busy(provider.backends(filters=lambda x: int(x.configuration().n\_qubits) >= 3 and

not x.configuration().simulator and x.status().operational==True))

print("Running on current least busy device: ", device)

# Run our circuit on the least busy backend. Monitor the execution of the job in the queue

from qiskit.tools.monitor import job\_monitor

transpiled\_grover\_circuit = transpile(grover\_circuit, device, optimization\_level=3)

job = device.run(transpiled\_grover\_circuit)

job\_monitor(job, interval=2)

# Get the results from the computation

results = job.result()

answer = results.get\_counts(grover\_circuit)

plot\_histogram(answer)

