ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΚΒΑΝΤΙΚΗ ΥΠΟΛΟΓΙΣΤΙΚΗ
-
ΣΥΓΚΡΙΣΗ QISKIT ΚΑΙ QUANTUM DEVELOPMENT KIT

ΚΛΕΙΣΑΡΧΑΚΗΣ ΔΗΜΗΤΡΙΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΝΙΚΗΤΑΣ-ΜΑΡΙΝΟΣ ΣΓΟΥΡΟΣ

ΠΕΙΡΑΙΑΣ

ΟΚΤΩΒΡΙΟΣ 2019

Περίληψη

Η παρούσα εργασία έχει ως θέμα της την Κβαντική Υπολογιστική καθώς και την περιγραφή και την σύγκριση μεταξύ των κιτ κβαντικού λογισμικού, Qiskit της IBM και Quantum Development Kit της Microsoft. Η εργασία χωρίζεται σε δύο κύρια μέρη.

Κατά το πρώτο μέρος της εργασίας (κεφάλαια 2-4) περιγράφονται όλες οι βασικές έννοιες της Κβαντικής Υπολογιστικής, οι οποίες αποτελούν και προϋπόθεση για την κατανόηση του δεύτερου μέρους της εργασίας. Αρχικά, περιγράφεται εν συντομία ο κλάδος της Κβαντικής Μηχανικής και παρουσιάζεται μία σύντομη ιστορική αναδρομή από την αρχή του μέχρι και την εμφάνιση της Κβαντικής Υπολογιστικής. Στην συνέχεια ακολουθεί μία περιγραφή των βασικών εννοιών της Κβαντικής Υπολογιστικής, παρουσιάζεται το απαραίτητο μαθηματικό υπόβαθρο για την κατανόηση των εννοιών αυτών και περιγράφονται βασικές ιδιότητες της Κβαντικής Μηχανικής, όπως η Κβαντική Υπέρθεση και η Κβαντική Διεμπλοκή, οι οποίες αξιοποιούνται σε μεγάλο βαθμό στην Κβαντική Υπολογιστική. Κατά το τέλος του πρώτου μέρους της εργασίας, παρουσιάζεται το Κβαντικό Μοντέλο Προγραμματισμού για την κατασκευή και την εκτέλεση κβαντικών κυκλωμάτων και περιγράφονται ο αλγόριθμος της Κβαντικής Τηλεμεταφοράς και ο αλγόριθμος Bernstein-Vazirani.

Κατά το δεύτερο μέρος της εργασίας (κεφάλαια 5-7) περιγράφονται τα δύο κιτ κβαντικού λογισμικού, Qiskit και Quantum Development Kit. Για το καθένα δίνεται μία σύντομη περιγραφή και παρουσιάζονται τα στοιχεία που αφορούν τη λειτουργία και το μοντέλο προγραμματισμού του με την χρήση παραδειγμάτων. Στη συνέχεια, εκτελούνται οι αλγόριθμοι της Κβαντικής Τηλεμεταφοράς και του αλγόριθμου Bernstein-Vazirani στο κάθε κιτ ξεχωριστά και εξετάζονται τα αποτελέσματά τους. Στο τελευταίο κεφάλαιο, παρουσιάζεται ένας πίνακας σύγκρισης μεταξύ των δύο κιτ κβαντικού λογισμικού.

Πίνακας Περιεχομένων

1 Εισαγωγή 5

1.1 Κβάντο 5

1.2 Κβαντική Μηχανική 5

1.3 Ιστορική αναδρομή 6

1.3.1 Ακτινοβολία Μέλανος Σώματος 6

1.3.2 Φωτοηλεκτρικό Φαινόμενο 6

1.3.3 Αστάθεια Ατόμου Rutherford 7

1.3.4 Κυματοσωματιδιακός Δυϊσμός της Ύλης 7

1.3.5 Εξίσωση του Schrodinger 7

1.3.6 Ανισότητες Bell 8

1.3.7 Κβαντική Υπολογιστική 8

1.4 Κβαντική Υπολογιστική 9

2 Βασικές Έννοιες 10

2.1 Κλασσικό Bit 10

2.2 Κβαντική Υπέρθεση 10

2.3 Κβαντικό Bit 10

2.3.1 Συμβολισμός Dirac 11

2.3.2 Φυσικές Υλοποιήσεις Qubit 13

2.4 Μέτρηση 13

2.4.1 Γεωμετρική Αναπαράσταση 14

2.4.2 Μετρήσεις σε Βάσεις Διαφορετικές από την Κανονική Βάση 15

2.4.3 Βάση Sign 16

2.5 Τανυστικό Γινόμενο Διανυσματικών Χώρων 17

2.5.1 Μερική Μέτρηση 18

2.5.2 Διαπλεγμένες Καταστάσεις 19

2.6 Κβαντική Διεμπλοκή 20

2.6.1 Καταστάσεις Bell 21

3 Κβαντικό Μοντέλο Προγραμματισμού 22

3.1 Κλασσικές Λογικές Πύλες 22

3.2 Κβαντικές Λογικές Πύλες 24

3.2.1 Κβαντικές Πύλες ενός Qubit 25
3.2.2 Κβαντικές Πύλες για δύο Qubits 28
3.3 Μοντέλο Προγραμματισμού Κβαντικών Κυκλωμάτων 29
3.4 Κβαντικοί Αλγόριθμοι 30
3.4.1 Δημιουργία Καταστάσεων Bell 30
3.4.2 Κβαντική Τηλεμεταφορά 32
3.4.3 Αλγόριθμός Bernstein-Vazirani 35
3.5 Κβαντικοί Υπολογιστές 40
3.6 Κβαντικές Γλώσσες Προγραμματισμού και Προσομοίωση Κβαντικών Κυκλωμάτων 41
4 Qiskit 42
4.1 Εισαγωγικά 42
4.2 Qiskit Elements 43
4.2.1 Terra 43
4.2.2 Aer 44
4.2.3 Ignis 44
4.2.4 Aqua 44
4.3 Αναπαράσταση Κβαντικών Καταστάσεων 45
4.4 Επισκόπηση Qiskit 45
4.4.1 Δημιουργία Κβαντικού Κυκλώματος 45
4.4.2 Υλοποίηση Κβαντικών Πυλών 46
4.4.3 Αναπαράσταση Κβαντικού Κυκλώματος 47
4.4.4 Μέτρηση Κβαντικού Κυκλώματος 47
4.4.5 Εκτέλεση Κβαντικού Κυκλώματος σε Προσομοιωτή 48
4.4.6 Απεικόνιση Αποτελεσμάτων σε Ιστόγραμμα 50
4.4.7 Εκτέλεση Κβαντικού Κυκλώματος σε Κβαντικό Υπολογιστή 51
4.5 Κβαντική Τηλεμεταφορά 55
4.6 Αλγόριθμός Bernstein-Vazirani 58
4.7 IBM Q Experience 64
4.7.1 Circuit Composer 64
4.7.2 Qiskit Notebooks 66
5 Quantum Development Kit 67
5.1 Εισαγωγικά 67
5.2 Γλώσσα Προγραμματισμού Q# 67

5.3 Quantum Katas 67
5.4 Μοντέλο προγραμματισμού Quantum Development Kit 68
5.5 Βασικοί Τύποι Μεταβλητών της Q# 68
5.6 Διανομή των Qubit 69
5.7 Mutable και Immutable μεταβλητές 69
5.8 Δημιουργία Κβαντικής Διεμπλοκής 70
5.9 Κβαντική Τηλεμεταφορά 73
5.10 Αλγόριθμός Bernstein-Vazirani 76
6 Πίνακας Σύγκρισης 80
7 Βιβλιογραφία 82

1 Εισαγωγή

1.1 Κβάντο

Η λέξη **κβάντο (quantum)** προέρχεται από τα λατινικά και σημαίνει «ποσότητα». Στη φυσική, αναπαριστά την ελάχιστη δυνατή διακριτή μονάδα μέτρησης ενός φυσικού μεγέθους το οποίο λαμβάνει διακριτές τιμές (ορισμός ο οποίος χρησιμοποιήθηκε πρώτη φορά το 1900 από τον φυσικό **Max Planck**). Συνεπώς, όλες οι ποσότητες αυτού του μεγέθους είναι πάντα ακέραια πολλαπλάσια αυτής της μονάδας και δεν μπορούν να υπάρξουν δεκαδικές ποσότητες.

Για παράδειγμα, το φως είναι μια ηλεκτρομαγνητική ακτινοβολία η οποία διαδίδεται σε μικρά διακριτά «πακέτα» ενέργειας, που ονομάζονται **κβάντα φωτός** ή **φωτόνια**. Το κβάντο φωτός (φωτόνιο) επομένως, είναι η μικρότερη δυνατή διακριτή ποσότητα φωτός που μπορεί να υπάρξει. Η ενέργεια αυτή είναι κβαντισμένη, δηλαδή μπορεί να πάρει μόνο ορισμένες διακριτές καθορισμένες τιμές. Παρομοίως, η ενέργεια ενός ηλεκτρονίου εντός ενός ατόμου είναι επίσης κβαντισμένη, δηλαδή μπορεί να πάρει μία εκ του συνόλου των επιτρεπόμενων τιμών ενέργειας (ενεργειακές στάθμες).

Η **κβάντιση (quantization)** αποτελεί θεμέλιο της **κβαντικής μηχανικής**. Η κβάντιση της ενέργειας και το πώς αυτό επηρεάζει την αλληλεπίδρασή της με την ύλη, αποτελεί μέρος του θεμελιώδους πλαισίου για την κατανόηση και την περιγραφή της φύσης.

1.2 Κβαντική Μηχανική

Η **Κβαντική Μηχανική** (ή Κβαντική Θεωρία), είναι η θεωρία που περιγράφει με ακρίβεια τις φυσικές διεργασίες σε επίπεδο μικρόκοσμου (διαστάσεις ατομικών ή υπό-ατομικών συστημάτων), χρησιμοποιώντας ένα σύνολο από υποθέσεις. Οι υποθέσεις αυτές πολλές φορές είναι ασύμβατες με τη σχετική με το μακρόκοσμο εμπειρία. Στις μικροσκοπικές αυτές διαστάσεις, η κλασσική φυσική αποτυγχάνει να περιγράψει με ακρίβεια τη δομή φυσικών συστημάτων (άτομα, μόρια, πυρήνες ατόμων, στοιχειώδη σωμάτια) καθώς και την χρονική τους εξέλιξη [1].

Ο όρος *κβαντική θεωρία* χρησιμοποιείται κατά την περίοδο 1900-1920 όταν όλα ακόμα βρίσκονταν σε επίπεδο απλών αξιωμάτων και τύπων που βοηθούσαν στην αποσαφήνιση ορισμένων ζητημάτων τα οποία αφορούσαν τη φύση του φωτός και τη δομή των ατόμων. Ο όρος *κβαντική μηχανική* αναφέρεται στην «επανάσταση» που συνέβη την δεκαετία του 1920 όταν η νέα αυτή κοσμοθεωρία αντικατέστησε τη μηχανική του Νεύτωνα με σκοπό να περιγράψει τη δομή του υποατομικού κόσμου [2].

Η βασική διαφοροποίηση της Κβαντικής από την Κλασσική Φυσική, συνοψίζεται στην **Αρχή της Αβεβαιότητας (Uncertainty Principle).** Στην Κλασσική Φυσική, η μόνη πηγή αβεβαιότητας προέρχεται από τα πιθανά σφάλματα των μετρήσεων. Βελτιώνοντας την ακρίβεια των πειραματικών μετρήσεων, είναι δυνατό τα σφάλματα αυτά στις κλασσικές μετρήσεις να μειωθούν και θεωρητικά να τείνουν στο μηδέν. Αντίθετα, στην Κβαντική Μηχανική η αβεβαιότητα ορισμένων μετρήσεων είναι ενδογενής στη θεωρία και στη φύση και δεν μπορεί να μειωθεί, όσο και αν βελτιωθεί η πειραματική διαδικασία. Η αβεβαιότητα αυτή οφείλεται στην **Μη Τοπικότητα (Quantum Nonlocality)**, που είναι ενδογενής αρχή της Κβαντικής

Μηχανικής. Σύμφωνα με αυτή, τα στοιχειώδη σωμάτια της φύσης δεν συμπεριφέρονται ως εντοπισμένα σημειακά σωμάτια σε ένα σημείο του χώρου, αλλά συμπεριφέρονται σαν να υπάρχουν ταυτόχρονα, σε πολλά σημεία του χώρου, μέχρι να γίνει η μέτρηση της θέσης τους [1]. Η χρονική τους εξέλιξη δεν περιγράφεται από μια τροχιά στον χώρο, αλλά από την εξέλιξη ***«πλατών πιθανότητας»***. Τα πλάτη πιθανότητας είναι μιγαδικές εν γένει συναρτήσεις, των οποίων το μέτρο στο τετράγωνο εκφράζει την πιθανότητα για την εμφάνιση διαφόρων αποτελεσμάτων των πειραματικών μετρήσεων [1].

1.3 Ιστορική αναδρομή

Ακολουθεί μία σύντομη περιγραφή ορισμένων βασικών σταθμών στην ιστορική εξέλιξη της Κβαντικής Μηχανικής.

1.3.1 Ακτινοβολία Μέλανος Σώματος

Το 1900 ο **Max Planck** διατύπωσε την υπόθεση ότι η φωτεινή ενέργεια που ακτινοβολείται από ένα θερμαινόμενο σώμα δεν εκπέμπεται κατά συνεχή ροή αλλά σε μορφή αυτοτελών ποσοτήτων ***'ε' (κβάντα ενέργειας)*** που είναι ανάλογες προς τη συχνότητα ***f*** του εκπεμπόμενου φωτός.

$$E = nhf = nhc/\lambda$$

- ***n***, ακέραιος μεγαλύτερος του μηδέν,
- $h \simeq 6.626 \times 10^{-34}\ Js$, η οποία είναι γνωστή ως *σταθερά του Planck*,
- ***f***, η συχνότητα του εκπεμπόμενου φωτός,
- ***c,*** η ταχύτητα του φωτός στο κενό,
- λ**,** το μήκος κύματος.

1.3.2 Φωτοηλεκτρικό Φαινόμενο

Το 1905 ο **Albert Einstein** επεκτείνοντας τις απόψεις του Planck, υπέθεσε ότι το φως αποτελείται από μικρά «**πακέτα» ενέργειας**, που ονομάζονται **κβάντα φωτός** ή **φωτόνια**. Η ενέργεια κάθε φωτονίου είναι:

$$E = hf$$

1.3.3 Αστάθεια Ατόμου Rutherford

Το 1911 ο **Ernest Rutherford**, έδειξε ότι τα άτομα αποτελούνται από συμπαγή, θετικά φορτισμένο πυρήνα και ηλεκτρόνια που κινούνται στην περιοχή του πυρήνα. Σύμφωνα με την κλασσική φυσική, τα άτομα της ύλης θα έπρεπε να είναι ασταθή διότι τα επιταχυνόμενα, λόγω της δύναμης Coulomb, ηλεκτρόνια εκπέμπουν ακτινοβολία και χάνουν ενέργεια, οπότε θα έπρεπε να πέφτουν στον πυρήνα [1]. Αντίθετα όμως με τα αναμενόμενα, τα άτομα είναι σταθερά.

Το 1913, ο **Niels Bohr** πρότεινε ένα μοντέλο, που είχε την δυνατότητα να ερμηνεύσει το φαινόμενο της ευστάθειας των ατόμων. Σύμφωνα με το μοντέλο αυτό, το ηλεκτρόνιο μπορεί να κινείται μόνο σε ορισμένες τροχιές, οι οποίες ονομάζονται **επιτρεπόμενες**. Οι επιτρεπόμενες τροχιές είναι εκείνες για τις οποίες ισχύει ότι η στροφορμή του ηλεκτρονίου είναι **κβαντισμένη** και ίση με ακέραιο πολλαπλάσιο της ποσότητας $\hbar = h/(2\pi)$. Οι επιτρεπόμενες τιμές ενέργειας ονομάζονται **ενεργειακές στάθμες**. Περαιτέρω, το μοντέλο Bohr προβλέπει την εκπομπή **«πακέτων» ακτινοβολίας**, κατά τη μετάβαση ηλεκτρονίων, από μια στάθμη υψηλότερης ενέργειας σε μια στάθμη χαμηλότερης ενέργειας.

1.3.4 Κυματοσωματιδιακός Δυϊσμός της Ύλης

Το 1924, στη διδακτορική του διατριβή ο **Louis de Broglie**, για την ερμηνεία της διακριτότητας της στροφορμής στο μοντέλο του Bohr, θεώρησε ότι όλα τα σωματίδια έχουν ταυτόχρονα και κυματική συμπεριφορά παράλληλα με τη σωματιδιακή. Κάθε σωματίδιο ενέργειας ***E*** και ορμής ***p*** συμπεριφέρεται και σαν ένα κύμα μήκους κύματος $\lambda = h/p$ και συχνότητας $f = E/h$.

1.3.5 Εξίσωση του Schrodinger

Το 1926, ο **Erwin Schrodinger**, σύμφωνα με τα παραπάνω, οδηγήθηκε στην διατύπωση της **κυματικής εξίσωσης**, πού περιγράφει την εξέλιξη ενός κβαντικού συστήματος, η οποία αποτελεί μια από τις βασικές αρχές της Κβαντικής Θεωρίας. Στην Κβαντική Μηχανική, η εξίσωση του Schrodinger έχει σημασία ανάλογη του δεύτερου νόμου του Νεύτωνα στην κλασσική μηχανική.

Τα κινούμενα μικροσκοπικά σωμάτια συμπεριφέρονται σαν κύματα. Η ποσότητα που ταλαντώνεται, είναι το πλάτος πιθανότητας εύρεσης ενός σωματίου σε μια θέση στο χώρο (**κυματοσυνάρτηση Ψ**). Το πλάτος αυτό είναι μιγαδική ποσότητα και υπακούει σε συγκεκριμένη κυματική εξίσωση, η οποία καθορίζει τη χρονική του εξέλιξη σε κάθε σημείο του χώρου. Η κυματοσυνάρτηση **Ψ** περιέχει όλη την πληροφορία η οποία μπορεί να συλλεχθεί για το σωμάτιο, το οποίο περιγράφει.

1.3.6 Ανισότητες Bell

Ο Einstein θεωρούσε τον πιθανοκρατικό χαρακτήρα της Κβαντικής Μηχανικής προβληματικό (από όπου προέρχεται και η γνωστή φράση του, "*God doesn't play dice*") και υποστήριζε ότι υπάρχει μια πιο θεμελιώδης και πλήρης θεωρία, η οποία μπορεί να απαλλάξει από την πρόβλεψη πιθανοτήτων (κυματοσυνάρτηση) και να την αντικαταστήσει με ακριβείς προβλέψεις, χρησιμοποιώντας νέες μεταβλητές, που είναι «κρυμμένες» (**hidden variables**).

Το 1964, ο **John Stewart Bell** διατύπωσε μαθηματικά, με την μορφή ανισοτήτων, τις ανισότητες Bell, θέλοντας να απαντήσει στο ερώτημα αν η αβεβαιότητα και η ύπαρξη κυματοσυνάρτησης στην Κβαντική Θεωρία είναι αποτέλεσμα ελλιπούς γνώσης της πραγματικής θεωρίας. Οι ανισότητες αυτές επαληθεύτηκαν το 1972 από τους **John Francis Clauser** και **Stuart Freedman** και αποδείχτηκε ότι δεν υπάρχει πιο πλήρης θεωρία από την Κβαντική Μηχανική και συνεπώς δεν υπάρχουν κρυμμένες μεταβλητές.

1.3.7 Κβαντική Υπολογιστική

Από τις δεκαετίες του '60 και '70 ξεκίνησε η διαμόρφωση της ιδέας της Κβαντικής Υπολογιστικής (Quantum Computing), αλλά δεν ήταν μέχρι την δεκαετία του '80 όταν ο τομέας αυτός άρχισε να εξαπλώνεται. Ορισμένα σημαντικά ορόσημα της δεκαετίας αυτής:

- Το 1980 ο **Paul Benioff** με την εργασία του «*The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines*» έδειξε ότι ένας υπολογιστής θα μπορούσε να λειτουργήσει σύμφωνα με τους νόμους της Κβαντικής Μηχανικής.
- Το 1982 ο **Richard Feynman** σε μία ομιλία του στο MIT παρατήρησε ότι ήταν αδύνατο να προσομοιωθεί αποτελεσματικά μια εξέλιξη ενός κβαντικού συστήματος σε έναν κλασσικό υπολογιστή και πρότεινε ένα βασικό μοντέλο για έναν «κβαντικό υπολογιστή».
- Το 1985 ο **David Deutsch** στο Πανεπιστήμιο της Οξφόρδης, περιγράφει τον πρώτο *Καθολικό Κβαντικό Υπολογιστή*. Ακριβώς όπως μια *Καθολική Μηχανή Turing* μπορεί να εξομοιώνει αποτελεσματικά οποιαδήποτε άλλη μηχανή Turing, έτσι ο *Καθολικός Κβαντικός Υπολογιστής* είναι σε θέση να προσομοιώσει οποιοδήποτε άλλο κβαντικό υπολογιστή με το πολύ πολυωνυμική βελτίωση.

1.4 Κβαντική Υπολογιστική

Με τον όρο **Κβαντική Υπολογιστική (Quantum Computing)** εννοείται η χρήση ορισμένων βασικών ιδιοτήτων της Κβαντικής Μηχανικής, όπως η **Κβαντική Υπέρθεση (Quantum Superposition)** και η **Κβαντική Διεμπλοκή (Quantum Entanglement)**, με σκοπό την αποτελεσματικότερη εκτέλεση ενός υπολογισμού. Σε σύγκριση με την κλασσική υπολογιστική, η Κβαντική Υπολογιστική προσφέρει τη δυνατότητα να μειωθεί δραστικά ο χρόνος εκτέλεσης και η κατανάλωση υπολογιστικών πόρων ορισμένων υπολογισμών. Πλέον, υπάρχει μια ευρεία συναίνεση ότι η **Κβαντική Υπεροχή (Quantum Supremacy)**, δηλαδή η ικανότητα των κβαντικών υπολογιστικών συσκευών να λύσουν προβλήματα που οι κλασσικοί υπολογιστές δεν μπορούν πρακτικά, είναι επικείμενη [3]. Η Κβαντική Υπολογιστική μπορεί μια μέρα να φέρει επανάσταση σε τομείς όπως η τεχνητή νοημοσύνη, η κρυπτογραφία, η φαρμακευτική έρευνα, η μοριακή μοντελοποίηση, τα οικονομικά συστήματα και κατ' επέκταση να βοηθήσει την ανθρωπότητα να διερευνήσει και να κατανοήσει καλύτερα το σύμπαν.

Η μαθηματική βάση της Κβαντικής Υπολογιστικής, το μοντέλο προγραμματισμού και οι περισσότεροι κβαντικοί αλγόριθμοι ξεκίνησαν να δημοσιεύονται από τη δεκαετία του 1990. Σε αντίθεση με παλαιότερα, θεωρείται ότι έχει έρθει η στιγμή να καταστούν οι κβαντικοί αλγόριθμοι και οι εφαρμογές τους προσιτοί σε ένα ευρύ φάσμα ερευνητών και προγραμματιστών, σε όλη την επιστήμη των υπολογιστών [3].

Το κβαντικό μοντέλο προγραμματισμού όμως είναι θεμελιωδώς διαφορετικό από τον παραδοσιακό προγραμματισμό υπολογιστών. Στα επόμενα δύο κεφάλαια θα περιγραφούν αναλυτικά όλες οι βασικές έννοιες της Κβαντικής Υπολογιστικής καθώς και το μοντέλο προγραμματισμού της.

2 Βασικές Έννοιες

2.1 Κλασσικό Bit

Το **bit** είναι η στοιχειώδης μονάδα πληροφορίας η οποία περιγράφει ένα διδιάστατο κλασσικό σύστημα (δηλαδή ένα σύστημα το οποίο μπορεί να υπάρχει σε μία από δύο διακριτές καταστάσεις). Παραδείγματα συστημάτων τα οποία μπορούν να εκφραστούν με ένα bit είναι:

- Ένας διακόπτης ανοιχτός ή κλειστός.
- Μία πρόταση "αληθής" ή "ψευδής".

Το bit είναι ένας τρόπος περιγραφής ενός συστήματος το οποίο αποτελείται συνολικά από δύο καταστάσεις οι οποίες συνήθως συμβολίζονται ως 0 και 1 (ή "false" και "true" αντίστοιχα). Επομένως ένα bit μπορεί να είναι είτε στην κατάσταση 0 είτε στην κατάσταση 1. Είτε ένας διακόπτης είναι ανοιχτός, είτε είναι κλειστός. Είτε μία πρόταση είναι αληθής, είτε είναι ψευδής.

Ένα bit μπορεί να αναπαρασταθεί ως ένας πίνακας 2×1.

- Η κατάσταση 0 συμβολίζεται ως $[1\ 0]$
- Η κατάσταση 1 συμβολίζεται ως $[0\ 1]$

Η πρώτη σειρά του πίνακα αντιπροσωπεύει την κατάσταση 0 του συστήματος και η δεύτερη σειρά του πίνακα αντιπροσωπεύει την κατάσταση 1.

2.2 Κβαντική Υπέρθεση

Σε αντίθεση με τα κλασσικά συστήματα, τα κβαντικά συστήματα μπορούν να βρίσκονται ταυτόχρονα και στις δύο πιθανές καταστάσεις (**Κβαντική Υπέρθεση – Quantum Superposition**), χωρίς η μία να αποκλείει την άλλη. Υπάρχουν δηλαδή συστήματα στα οποία ο αντίστοιχος διακόπτης θα ήταν την ίδια στιγμή ανοιχτός και κλειστός. Ένα κβαντικό σύστημα μπορεί να είναι στην κατάσταση 0 και στην κατάσταση 1 ταυτόχρονα. Προκύπτει έτσι η ανάγκη για μία διαφορετική μονάδα πληροφορίας η οποία θα περιγράφει ένα διδιάστατο κβαντικό σύστημα.

2.3 Κβαντικό Bit

Το **κβαντικό bit** (ή **qubit**) είναι η στοιχειώδης μονάδα πληροφορίας που χρησιμοποιείται για την περιγραφή ενός διδιάστατου κβαντικού συστήματος. Ένα qubit, όπως και ένα bit, παριστάνεται ως ένας πίνακας 2 × 1 αλλά με μιγαδικούς αριθμούς για τιμές [4].

$$[\alpha\ \beta]$$

Οι τιμές $\boldsymbol{\alpha}$ και $\boldsymbol{\beta}$ είναι μιγαδικοί αριθμοί ($\alpha, \beta \in \mathbb{C}$) και καλούνται **πλάτη πιθανότητας**. Το πλάτος πιθανότητας μπορεί να χαρακτηριστεί ως το «μέτρο της κλίσης του συστήματος» προς μια κατάσταση. Επομένως, το σύστημα βρίσκεται με μια κλίση $\boldsymbol{\alpha}$ προς την κατάσταση 0 και με μία κλίση $\boldsymbol{\beta}$ προς την κατάσταση 1, δεδομένου ότι ένα κβαντικό σύστημα βρίσκεται ταυτόχρονα και στις δύο καταστάσεις. Το

πλάτος πιθανότητας δεν πρέπει να συγχέεται με την πιθανότητα του να βρίσκεται το σύστημα σε μία από τις δύο καταστάσεις. Τα πλάτη πιθανότητας μπορεί να είναι αρνητικά ή/και φανταστικά.

Το τετράγωνο του μέτρου του πλάτους πιθανότητας είναι αυτό που αντιπροσωπεύει την πιθανότητα εμφάνισής του συστήματος σε μία συγκεκριμένη κατάσταση. Δηλαδή:

- με πιθανότητα $|\alpha|^2$ το σύστημα θα βρίσκεται στην κατάσταση 0.
- με πιθανότητα $|\beta|^2$ το σύστημα θα βρίσκεται στην κατάσταση 1.

Σύμφωνα με τα παραπάνω, καθίσταται αναγκαίο η **κβαντική κατάσταση (quantum state)** του συστήματος να είναι **κανονικοποιημένη (normalized)**, διότι οι πιθανότητες πρέπει να έχουν άθροισμα ίσο με 1. Συνεπώς:

$$|\alpha|^2 + |\beta|^2 = 1$$

Δεδομένου ότι $\boldsymbol{\alpha}$ και $\boldsymbol{\beta}$ είναι μιγαδικοί αριθμοί, συμβολίζονται ως εξής:

$$\alpha = \alpha_0 + i\alpha_1$$

$$\beta = \beta_0 + i\beta_1$$

Τα μέτρα των μιγαδικών δίνονται από τον ακόλουθο τύπο:

$$|\alpha| = \sqrt{\alpha_0^2 + \alpha_1^2}$$

$$|\beta| = \sqrt{\beta_0^2 + \beta_1^2}$$

Επομένως, πρέπει:

$$\left(\alpha_0^2 + \alpha_1^2\right) + \left(\beta_0^2 + \beta_1^2\right) = 1$$

Η **κανονικοποίηση (normalization)** δείχνει ότι η κβαντική κατάσταση του συστήματος είναι ένα **μοναδιαίο διάνυσμα (unit vector)** (διάνυσμα με μέτρο ίσο με 1) σε ένα μιγαδικό διανυσματικό χώρο δύο διαστάσεων.

2.3.1 Συμβολισμός Dirac

Στην Κβαντική Μηχανική, ο **συμβολισμός Dirac (bra-ket notation)** χρησιμοποιείται για την περιγραφή των κβαντικών καταστάσεων (quantum states).

Ένα **διάνυσμα στήλης (column vector)** συμβολίζεται ως “ $|\psi\rangle$ ” και ονομάζεται **ket**. Ως διάνυσμα στήλης ορίζεται ένας n × 1 πίνακας αποτελούμενος δηλαδή από μία στήλη n στοιχείων.

Η **κατάσταση 0** ή αλλιώς **θεμελιώδης κατάσταση (ground state)** ενός κβαντικού συστήματος επομένως, συμβολίζεται ως:

$$|0\rangle = [1\ 0\ ]$$

Αντίστοιχα, η **κατάσταση 1** ή αλλιώς **διεγερμένη κατάσταση (excited state)** ενός κβαντικού συστήματος συμβολίζεται ως:

$$|1\rangle = [0\ 1\ ]$$

Μία τυχαία κατάσταση ενός κβαντικού συστήματος δύο καταστάσεων συμβολίζεται ως εξής:

$$|\psi\rangle = [\alpha\ \beta\ ]$$

Όπου, $\alpha, \beta \in \mathbb{C}$ και $|\alpha|^2 + |\beta|^2 = 1$.

Ισοδύναμα, η παραπάνω κατάσταση $|\psi\rangle$ μπορεί να γραφτεί ως:

$$|\psi\rangle = \alpha\,[1\ 0\ ] + \beta\,[0\ 1\ ] = \alpha|0\rangle + \beta|1\rangle$$

Γενικότερα, για ένα σύστημα *k* καταστάσεων, μία κβαντική κατάσταση $|\psi\rangle$ περιγράφεται από μια ακολουθία *k* μιγαδικών αριθμών $\alpha_0, \alpha_1, \ldots, \alpha_{\kappa-1} \in C$, έτσι ώστε $\sum_j |\alpha_j|^2 = 1$ (κανονικοποιημένη κβαντική κατάσταση).

Η κβαντική κατάσταση του συστήματος μπορεί να γραφτεί ως ένα διάνυσμα στήλης *k* διαστάσεων:

$$|\psi\rangle = \begin{bmatrix} \alpha_0 & \vdots & \alpha_{\kappa-1} \end{bmatrix} = \alpha_0|0\rangle + \ldots + \alpha_{\kappa-1}|\kappa-1\rangle$$

Επομένως, η κβαντική κατάσταση του συστήματος είναι ένα μοναδιαίο διάνυσμα (unit vector) σε ένα μιγαδικό διανυσματικό χώρο *k* διαστάσεων **(χώρος Hilbert).** Ο **χώρος Hilbert (Hilbert space)** αποτελεί μία γενίκευση της έννοιας του Ευκλείδειου Χώρου. Επεκτείνει τις μεθόδους της γραμμικής άλγεβρας και του λογισμού από το διδιάστατο Ευκλείδειο επίπεδο και τριδιάστατο χώρο σε χώρους με κάθε πεπερασμένο ή άπειρο αριθμό διαστάσεων.

Οι *k* διαφορετικές κλασσικές καταστάσεις του κβαντικού συστήματος, $|0\rangle, |1\rangle, \ldots, |\kappa-1\rangle$, σχηματίζουν μία **ορθοκανονική βάση (orthonormal basis)** (όλα τα διανύσματα είναι μοναδιαία και κάθετα μεταξύ τους) για την κβαντική κατάσταση $|\psi\rangle$. Όπου:

$$|0\rangle = \begin{bmatrix} 1\ 0\ 0 & \vdots & 0 \end{bmatrix},\ |1\rangle = \begin{bmatrix} 0\ 1\ 0 & \vdots & 0 \end{bmatrix}, \ldots,\ |\kappa-1\rangle = \begin{bmatrix} 0\ 0\ 0 & \vdots & 1 \end{bmatrix}$$

Συγκεκριμένα οι *k* διαφορετικές κλασσικές καταστάσεις του κβαντικού συστήματος, $|0\rangle, |1\rangle, \ldots, |\kappa - 1\rangle$, σχηματίζουν την **κανονική βάση (standard basis).** Κανονική βάση ενός διανυσματικού χώρου είναι η βάση που έχει ως διανύσματα τις στήλες του **μοναδιαίου πίνακα (identity matrix)** I_k. Όπου:

$$I_k = \begin{bmatrix} 1 & 0 & \ldots & 0 & 0 & 1 & \ldots & 0 & \vdots & 0 & \vdots & 0 & \ddots & \ldots & 0 & 1 \end{bmatrix}$$

2.3.2 Φυσικές Υλοποιήσεις Qubit

Για την φυσική υλοποίηση ενός bit χρησιμοποιούνται συστήματα δύο καταστάσεων όπως:

- οι δύο στάσεις ενός δισταθούς πολυδονητή (flip-flop)
- οι δύο θέσεις ενός διακόπτη
- οι δύο τάσεις ηλεκτρικού ρεύματος που επιτρέπονται σε ένα κύκλωμα
- οι δύο κατευθύνσεις μαγνητισμού

Αντίστοιχα, για την φυσική υλοποίηση ενός qubit χρησιμοποιούνται κβαντικά συστήματα δύο καταστάσεων όπως:

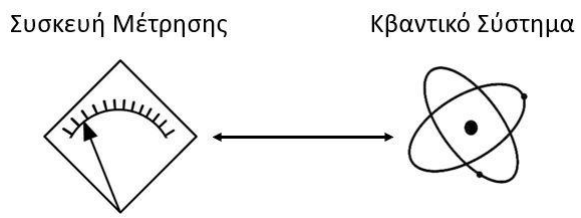
- οι δύο τροχιές ενός ηλεκτρονίου γύρω από τον πυρήνα ενός ατόμου (θεμελιώδης ή διεγερμένη κατάσταση)
- οι δύο πολωμένες καταστάσεις ενός φωτονίου (οριζόντια ή κάθετα)
- οι δύο τιμές του spin ενός υποατομικού σωματιδίου (πάνω ή κάτω)

2.4 Μέτρηση

Ένα από τα πιο δύσκολα και αμφιλεγόμενα προβλήματα στην Κβαντική Μηχανική αποτελεί το **πρόβλημα της μέτρησης (measurement problem).** Σε αντίθεση με τα κλασσικά συστήματα, όπου είναι δυνατή η εξέταση του κλασσικού bit προκειμένου να διαπιστωθεί αν βρίσκεται στην κατάσταση 0 ή στην κατάσταση 1, στα κβαντικά συστήματα είναι αδύνατο να πραγματοποιηθεί εξέταση ενός qubit, με την έννοια ότι είναι αδύνατη η ανάκτηση των πλατών πιθανότητας α και β. Στην κβαντική μηχανική η πληροφορία που μπορεί να ανακτηθεί για μία κβαντική κατάσταση είναι περιορισμένη.

Μία **μέτρηση (measurement)** ενός qubit δίνει ως αποτέλεσμα μία μόνο από τις δύο πιθανές καταστάσεις, ακόμα και αν αυτό βρισκόταν σε *υπέρθεση* ($\alpha|0\rangle + \beta|1\rangle$). Το αποτέλεσμα της μέτρησης θα είναι είτε η κατάσταση $|0\rangle$ με πιθανότητα $|\alpha|^2$ είτε η κατάσταση $|1\rangle$ με πιθανότητα $|\beta|^2$. Η ίδια η διαδικασία της μέτρησης δηλαδή, μεταβάλλει την κατάσταση του qubit, καταρρέοντας το από την υπέρθεσή του μεταξύ των καταστάσεων $|0\rangle$ και $|1\rangle$, στην κατάσταση η οποία είναι σύμφωνη με την προκύπτουσα κατάσταση της μέτρησης [5]. Για παράδειγμα, αν το αποτέλεσμα της μέτρησης ενός qubit, το οποίο βρίσκεται στην κατάσταση $\alpha|0\rangle + \beta|1\rangle$, είναι η κατάσταση $|0\rangle$, τότε το qubit μετά την μέτρηση θα βρίσκεται στην κατάσταση $|0\rangle$. Είναι αδύνατη επομένως η άντληση πληροφοριών για τα πλάτη πιθανότητας με επανάληψη της μέτρησης στο συγκεκριμένο qubit.

Είναι φανερό ότι για την πραγματοποίηση μίας μέτρησης χρειάζεται ένα κομμάτι εξοπλισμού μακροσκοπικού μεγέθους το οποίο συμπεριφέρεται σύμφωνα με τους νόμους της κλασσικής φυσικής. Για κάθε κβαντική μέτρηση χρειάζεται ένα κβαντικό σύστημα (μικροσκοπικού μεγέθους) του οποίου τα περιεχόμενα πρόκειται να εξετασθούν και μία **συσκευή (apparatus)** η οποία αλληλεπιδρά με το κβαντικό σύστημα. Η έξοδος (output) της συσκευής μπορεί να είναι ένας δείκτης σε ένα καντράν (dial) πάνω στην συσκευή ο οποίος θα δείχνει στο τελικό αποτέλεσμα της μέτρησης.



Εικόνα 2-1. Συσκευή μέτρησης κβαντικού συστήματος

Η συσκευή μέτρησης θα μπορούσε αντίστοιχα να παράγει ένα ηλεκτρικό σήμα σε ένα παλμογράφο ή μία ακολουθία bit αποθηκευμένη σε μνήμη υπολογιστή ή μία λάμψη φωτός πάνω σε μια φθορίζουσα οθόνη. Για λόγους ευκολίας, χρησιμοποιείται συχνά η απλή μορφή της συσκευής μέτρησης με δείκτη για την κατανόηση των μετρήσεων.

Γενικά, μία μέτρηση σε ένα κβαντικό σύστημα *κ* καταστάσεων δίνει ως αποτέλεσμα ένα μόνο από τα *κ* πιθανά αποτελέσματα, δηλαδή έναν ακέραιο αριθμό μεταξύ 0 και κ-1. Πραγματοποιώντας μία μέτρηση στην κατάσταση $|\psi\rangle$ στην κανονική βάση (standard basis), ο δείκτης της συσκευής μέτρησης δείχνει ένα αριθμό j, ο οποίος αντιστοιχεί στην κατάσταση $|j\rangle$, με πιθανότητα εμφάνισης $|\alpha_j|^2$ [6]. Συνεπώς:

$$P\,[j] = |\alpha_j|^2$$

Είναι ήδη γνωστό ότι $\sum_j |\alpha_j|^2 = 1$, διότι όλες οι πιθανότητες πρέπει να έχουν άθροισμα ίσο με 1.

2.4.1 Γεωμετρική Αναπαράσταση

Μία κατάσταση $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ μπορεί να αναπαρασταθεί ως ένα μοναδιαίο διάνυσμα σε ένα μιγαδικό διανυσματικό χώρο δύο διαστάσεων. Όπου, $\alpha, \beta \in \mathbb{C}$ και $|\alpha|^2 + |\beta|^2 = 1$.

Μία τυχαία κατάσταση $|\psi\rangle$ ως διάνυσμα στήλης δίνεται ως εξής:

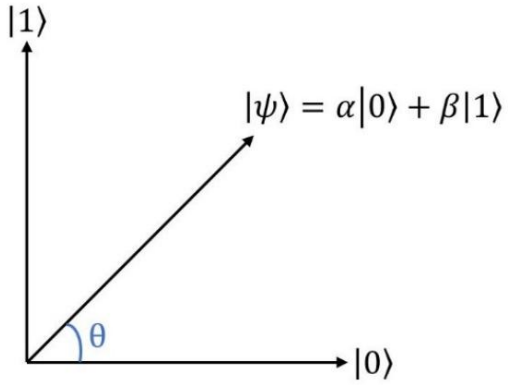
$$|\psi\rangle = [\alpha\ \beta\ ]$$

Αντίστοιχα, για τις καταστάσεις $|0\rangle$ και $|1\rangle$ προκύπτει ότι:

$$|0\rangle = [1\ 0\ ] \Rightarrow \alpha = 1,\ \beta = 0$$

$$|1\rangle = [0\ 1\ ] \Rightarrow \alpha = 0,\ \beta = 1$$

Για την ευκολότερη αναπαράσταση των κβαντικών καταστάσεων, γίνεται η υπόθεση ότι τα πλάτη πιθανότητας α και β είναι πραγματικοί αριθμοί. Δηλαδή, α, β ϵ ℝ.



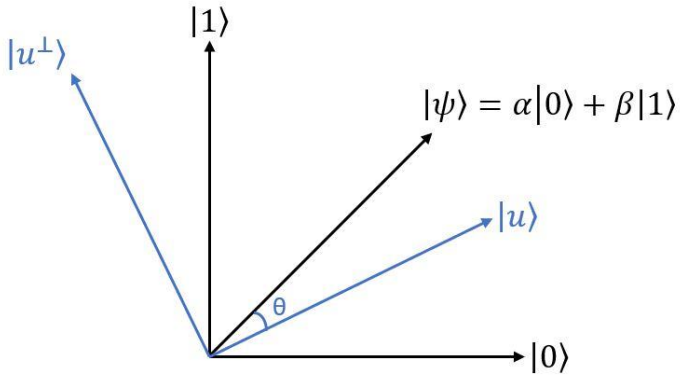
Εικόνα 2-2. Διάγραμμα κβαντικής κατάστασης $|\psi\rangle$

Από το παραπάνω διάγραμμα παρατηρείται ότι η κατάσταση $|0\rangle$ βρίσκεται πάνω στον άξονα x και η κατάσταση $|1\rangle$ βρίσκεται πάνω στον άξονα y. Μία τυχαία κατάσταση $|\psi\rangle$ προβάλλεται πάνω στις καταστάσεις $|0\rangle$ και $|1\rangle$, στην κανονική βάση στην οποία πραγματοποιείται η μέτρηση, με πιθανότητα $cos^2\theta$, όπου θ η γωνία που σχηματίζει η τυχαία κατάσταση $|\psi\rangle$ με την αντίστοιχη κατάσταση. Επομένως, μία μέτρηση είναι μία προβολή πάνω στην κανονική βάση [6]. Γεωμετρικά, προκύπτει ότι:

- η πιθανότητα μέτρησης της κατάστασης $|0\rangle$ είναι $cos^2\theta$.
- η πιθανότητα μέτρησης της κατάστασης $|1\rangle$ είναι $cos^2\left(\frac{\pi}{2} - \theta\right) = sin^2\theta$.

2.4.2 Μετρήσεις σε Βάσεις Διαφορετικές από την Κανονική Βάση

Η παραπάνω έννοια της μέτρησης μπορεί να γενικευτεί. Γενικά, μία μέτρηση είναι μία προβολή πάνω σε μία ορθοκανονική βάση. Αντί η μέτρηση της κβαντικής κατάστασης να γίνει στην κανονική βάση, μπορεί να επιλεχθεί οποιαδήποτε άλλη ορθοκανονική βάση $u,\ u^{\perp}$.



Εικόνα 2-3. Διάγραμμα κβαντικής κατάστασης $|\psi\rangle$ σε διαφορετική βάση

Παρομοίως, προκύπτει ότι:

- η πιθανότητα μέτρησης της κατάστασης $|u\rangle$ είναι $cos^2\theta$
- η πιθανότητα μέτρησης της κατάστασης $|u^{\perp}\rangle$ είναι $sin^2\theta$

όπου θ η γωνία που σχηματίζει η κατάσταση $|\psi\rangle$ με την κατάσταση $|u\rangle$.

Στην συγκεκριμένη περίπτωση η μέτρηση δεν γίνεται προκειμένου να διαπιστωθεί αν το qubit βρίσκεται στην θεμελιώδη ή διεγερμένη κατάσταση ($|0\rangle$ και $|1\rangle$ αντίστοιχα), αλλά αν βρίσκεται στην κατάσταση $|u\rangle$ ή $|u^{\perp}\rangle$.

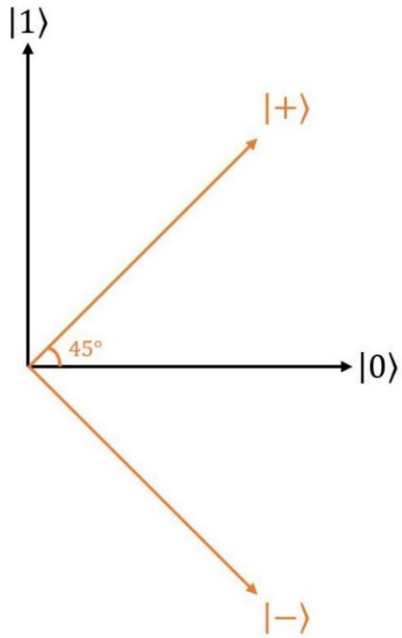
2.4.3 Βάση Sign

Μία σημαντική βάση, πέρα από την κανονική βάση είναι η **βάση sign (sign basis)**. Σχηματίζει μία γωνία $45°$ με την κανονική βάση.

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

$$|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

Σχηματικά:



Εικόνα 2-4. Βάση Sign

Προκύπτει ότι:

- η πιθανότητα μέτρησης της κατάστασης $|+\rangle$ είναι cos^2 (45°) = 1/2
- η πιθανότητα μέτρησης της κατάστασης $|-\rangle$ είναι sin^2 (45°) = 1/2

2.5 Τανυστικό Γινόμενο Διανυσματικών Χώρων

Στην Κβαντική Μηχανική δεν μελετώνται πάντα απλά και απομονωμένα σωματίδια. Στα πλαίσια της Κβαντικής Υπολογιστικής είναι αναγκαία η χρήση πολυπλοκότερων συστημάτων. Η μαθηματική κατανόηση τέτοιων συνθέτων συστημάτων προϋποθέτει την κατασκευή ενός χώρου Hilbert, ο οποίος είναι το προϊόν της συνθέσεως των επιμέρους χώρων Hilbert των διαφόρων σωματιδίων. Το μαθηματικό εργαλείο που επιτρέπει την κατασκευή αυτή ονομάζεται **τανυστικό γινόμενο (tensor product)** ή αλλιώς γινόμενο Kronecker [7]. Το τανυστικό γινόμενο επομένως, είναι η θεμελιώδης λειτουργία για την κατασκευή κβαντικών συστημάτων.

Έστω ότι το κβαντικό σύστημα αποτελείται από δύο σωματίδια και κατ' επέκταση δύο χώρους Hilbert, H_1 και H_2, έναν για κάθε σωματίδιο. Ο πρώτος έχει διάσταση N_1 και ο δεύτερος N_2. Ο σύνθετος χώρος H που αφορά το σύνθετο σύστημα, κατασκευάζεται από το τανυστικό γινόμενο. Το τανυστικό γινόμενο περιγράφει και τα δύο κβαντικά συστήματα ως ένα.

$$H = H_1 \otimes H_2$$

Η διάσταση του H είναι το γινόμενο των διαστάσεων του H_1 και H_2, δηλαδή:

$$\dim dim\ (H) = N_1 \times N_2$$

όπου N_1 η διάσταση του χώρου H_1 και αντίστοιχα N_2 η διάσταση του χώρου H_2.

Το τανυστικό γινόμενο μεταξύ δύο διανυσμάτων δίνεται ως εξής:

$$[\alpha\ \beta\ ] \otimes [\gamma\ \delta\ \varepsilon\ ] = [\alpha \times [\gamma\ \delta\ \varepsilon\ ]\ \beta \times [\gamma\ \delta\ \varepsilon\ ]\ ] = [\alpha\gamma\ \alpha\delta\ \alpha\varepsilon\ \beta\gamma\ \beta\delta\ \beta\varepsilon\ ] \quad [4]$$

Η διάσταση του σύνθετου χώρου Η των δύο διανυσμάτων δίνεται από το γινόμενο των διαστάσεων των δύο αρχικών χώρων.

$$\dim dim\ (H) = 2 \times 3 = 6$$

Έστω δύο qubit, το πρώτο στην κατάσταση $|0\rangle$ και το δεύτερο στην κατάσταση $|1\rangle$. Ο συνδυασμός τους γράφεται ως εξής:

$$|0\rangle \otimes |1\rangle = [1\ 0\ ] \otimes [0\ 1\ ] = [1 \times [0\ 1\ ]\ 0 \times [0\ 1\ ]\ ] = [0\ 1\ 0\ 0\ ]$$

Κατά σύμβαση, ο συνδυασμός των δύο καταστάσεων γράφεται χωρίς το σύμβολο του τανυστικού γινομένου « ⊗ » ως ένα ket με τον παρακάτω τρόπο.

$$|0\rangle \otimes |1\rangle = |0\rangle\, |1\rangle = |01\rangle$$

Έστω δύο κβαντικά συστήματα.

$$|\psi_0\rangle = \alpha_0|0\rangle + \beta_0|1\rangle \in C^2\,(H_1)$$

$$|\psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle \in C^2\,(H_2)$$

Ο συνδυασμός των δύο αυτών συστημάτων βρίσκεται σε ένα μιγαδικό διανυσματικό χώρο τεσσάρων διαστάσεων.

$$(N_1 \times N_2 = 2 \times 2 = 4)$$

Το τανυστικό γινόμενο εφαρμόζεται για κάθε ζεύγος καταστάσεων των δύο συστημάτων.

$$|\psi_0\rangle \otimes |\psi_1\rangle = (\alpha_0|0\rangle + \beta_0|1\rangle) \otimes (\alpha_1|0\rangle + \beta_1|1\rangle) =$$

$$\alpha_0\alpha_1(|0\rangle \otimes |0\rangle) + \alpha_0\beta_1(|0\rangle \otimes |1\rangle) + \beta_0\alpha_1(|1\rangle \otimes |0\rangle) + \beta_0\beta_1(|1\rangle \otimes |1\rangle) =$$

$$\gamma_{00}|00\rangle + \gamma_{01}|01\rangle + \gamma_{10}|10\rangle + \gamma_{11}|11\rangle \in C^4\,(H)$$

Ο σύνθετος μιγαδικός χώρος προκύπτει από το τανυστικό γινόμενο των δύο αρχικών μιγαδικών χώρων H_1 και H_2.

$$C^2 \otimes C^2 = C^4 = H$$

2.5.1 Μερική Μέτρηση

Σε κβαντικά συστήματα με παραπάνω από ένα qubit, δεν είναι πάντοτε απαραίτητο η μέτρηση να πραγματοποιείται σε όλα τα διαθέσιμα qubits. Πολλές φορές η μέτρηση πραγματοποιείται μόνο σε ένα ή κάποια από τα qubits.

Έστω ένα σύστημα με δύο qubits στην κατάσταση $|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$ στο οποίο πραγματοποιείται μία μέτρηση στο πρώτο qubit.

Η νέα υπέρθεση λαμβάνεται με την διαγραφή όλων των όρων της κατάστασης $|\psi\rangle$ που δεν είναι συμβατοί με το αποτέλεσμα της μέτρησης. Αν για παράδειγμα το αποτέλεσμα της μέτρησης του πρώτου qubit είναι η κατάσταση $|1\rangle$ τότε η νέα υπέρθεση που προκύπτει θα περιλαμβάνει μόνο εκείνους τους όρους όπου το πρώτο qubit είναι στην κατάσταση $|1\rangle$ [6] . Στο συγκεκριμένο παράδειγμα οι όροι αυτοί είναι οι $\alpha_{10}|10\rangle$ και $\alpha_{11}|11\rangle$.

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

Διαγράφοντας τους όρους οι οποίοι δεν είναι συμβατοί με το αποτέλεσμα της μέτρησης, το άθροισμα των τετραγώνων των πλατών πιθανότητας της κατάστασης δεν είναι πλέον ίσο 1. Επομένως, η νέα κατάσταση κανονικοποιείται διαιρώντας με την τετραγωνική ρίζα της πιθανότητας με την οποία μετρήθηκε το αποτέλεσμα.

$$|\psi\rangle = \frac{\alpha_{10}|10\rangle + \alpha_{11}|11\rangle}{\sqrt{|\alpha_{10}|^2 + |\alpha_{11}|^2}}$$

2.5.2 Διαπλεγμένες Καταστάσεις

Για μία σύνθετη κατάσταση είναι εφικτή η αποδόμησή της στις αρχικές της καταστάσεις οι οποίες την συνέθεταν. Για παράδειγμα έστω η εξής κατάσταση:

$$\frac{1}{2\sqrt{2}}|00\rangle - \frac{\sqrt{3}}{2\sqrt{2}}|01\rangle + \frac{1}{2\sqrt{2}}|10\rangle - \frac{\sqrt{3}}{2\sqrt{2}}|11\rangle$$

Με παραγοντοποίηση είναι εύκολο να ανακτηθούν οι δύο αρχικές καταστάσεις.

$$\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{2}|0\rangle - \frac{\sqrt{3}}{2}|1\rangle\right)$$

Υπάρχουν όμως σύνθετες καταστάσεις για τις οποίες είναι αδύνατο να γραφτούν ως τανυστικό γινόμενο δύο αρχικών καταστάσεων [6]. Για παράδειγμα έστω η εξής κατάσταση:

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Έστω οι δύο αρχικές καταστάσεις:

$$\alpha_0|0\rangle + \beta_0|1\rangle$$

$$\alpha_1|0\rangle + \beta_1|1\rangle$$

Συνθέτοντας τις αρχικές καταστάσεις:

$$(\alpha_0|0\rangle + \beta_0|1\rangle) \otimes (\alpha_1|0\rangle + \beta_1|1\rangle) =$$

$$\alpha_0\alpha_1|00\rangle + \alpha_0\beta_1|01\rangle + \beta_0\alpha_1|10\rangle + \beta_0\beta_1|11\rangle$$

Από την σύγκριση τελεστών προκύπτουν τα εξής:

$$\alpha_0\alpha_1 = \frac{1}{\sqrt{2}}$$

$$\alpha_0\beta_1 = 0$$

$$\beta_0\alpha_1 = 0$$

$$\beta_0\beta_1 = \frac{1}{\sqrt{2}}$$

Από τις προϋποθέσεις $\alpha_0 \bullet \alpha_1 = \frac{1}{\sqrt{2}}$, $\beta_0 \bullet \beta_1 = \frac{1}{\sqrt{2}}$ απαιτείται ότι α_0, α_1, β_0, $\beta_1 \neq 0$, γεγονός το οποίο δεν είναι συμβατό με τις προϋποθέσεις $\alpha_0 \bullet \beta_1 = 0$, $\beta_0 \bullet \alpha_1 = 0$. Επομένως, είναι αδύνατη η σύνθεση από τις αρχικές καταστάσεις. Οι καταστάσεις αυτές για τις οποίες είναι αδύνατο να γραφτούν ως τανυστικό γινόμενο των αρχικών καταστάσεων, αποτελούν και τις πιο ενδιαφέρουσες διότι αποτελούν ένα από τα πιο βασικά εργαλεία της Κβαντικής Υπολογιστικής. Αυτές οι καταστάσεις ονομάζονται **διαπλεγμένες καταστάσεις (entangled states).**

2.6 Κβαντική Διεμπλοκή

Η **κβαντική διεμπλοκή (quantum entanglement)** είναι μία από τις πιο χαρακτηριστικές ιδιότητες της Κβαντικής Μηχανικής η οποία αξιοποιείται σε πολύ μεγάλο βαθμό στην Κβαντική Υπολογιστική. Η κβαντική διεμπλοκή είναι ένα φαινόμενο στο οποίο οι κβαντικές καταστάσεις δύο ή περισσοτέρων σωματιδίων πρέπει να περιγράφονται αναφορικά με το άλλο και δεν μπορούν να περιγραφούν ανεξάρτητα από τις καταστάσεις των υπολοίπων, ακόμα και όταν τα σωματίδια διαχωρίζονται από μεγάλη απόσταση.

Σε ένα ζεύγος διαπλεγμένων qubits, πραγματοποιώντας μία μέτρηση στο πρώτο qubit, αυτόματα θα αλλάξει και η κατάσταση του δεύτερου qubit ανάλογα με την αλλαγή της κατάστασης του πρώτου qubit ως συνέπεια της μέτρησης.

Για παράδειγμα, έστω ότι το σύστημα βρίσκεται στην εξής κατάσταση:

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Τα δύο qubit, όπως έχει επιβεβαιωθεί προηγουμένως, είναι διαπλεγμένα. Επομένως, αν πραγματοποιηθεί μία μέτρηση στο πρώτο qubit και βρεθεί στην κατάσταση $|0\rangle$ τότε αυτόματα είναι γνωστό ότι και η κατάσταση του δεύτερου qubit θα είναι εξίσου η κατάσταση $|0\rangle$, χωρίς να χρειαστεί να πραγματοποιηθεί μία μέτρηση και για το δεύτερο qubit. Παρομοίως, αν το πρώτο qubit μετά την μέτρηση του βρεθεί στην κατάσταση $|1\rangle$ τότε είναι γνωστό ότι και η κατάσταση του δεύτερου qubit βρίσκεται εξίσου στην κατάσταση $|1\rangle$.

2.6.1 Καταστάσεις Bell

Οι καταστάσεις Bell είναι τέσσερις συγκεκριμένες διαπλεγμένες κβαντικές καταστάσεις με δύο qubits και αποτελούν τα απλούστερα παραδείγματα της κβαντικής διεμπλοκής.

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle$$

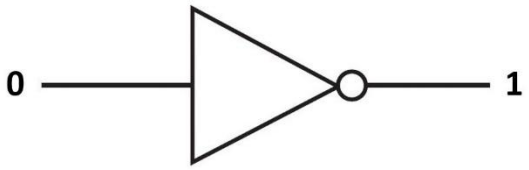
$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle$$

Λόγω της διεμπλοκής, η μέτρηση ενός qubit θα αποδώσει μία από τις δυο πιθανές τιμές στο άλλο qubit αμέσως, όπου η τιμή που αποδίδεται εξαρτάται από την κατάσταση Bell που βρίσκονται τα δύο qubits. Για παράδειγμα, αν τα δύο qubits βρίσκονται στην κατάσταση $|\Psi^+\rangle$ και η μέτρηση στο πρώτο qubit δώσει την κατάσταση $|1\rangle$ τότε είναι βέβαιο ότι το δεύτερο qubit βρίσκεται στην κατάσταση $|0\rangle$.

3 Κβαντικό Μοντέλο Προγραμματισμού

3.1 Κλασσικές Λογικές Πύλες

Οι κλασσικές λογικές πύλες είναι μαθηματικές οντότητες που αντιστοιχούν σε ηλεκτρονικά κυκλώματα, όπου εφαρμόζονται ένα ή περισσότερα σήματα εισόδου και τα οποία παράγουν ένα σήμα εξόδου. Τα ηλεκτρονικά σήματα, όπως οι τάσεις και τα ρεύματα, είναι αναλογικά σήματα που παίρνουν τιμές σε ένα δεδομένο εύρος όπως π.χ. 0 έως 3V. Σε ένα ψηφιακό σύστημα, θεωρείται ότι όλες οι τιμές σημάτων, οι οποίες περιλαμβάνονται σε ένα από τα δύο διακριτά προκαθορισμένα διαστήματα τιμών, αντιστοιχούν σε μία από τις δύο διακριτές λογικές τιμές 0 και 1. Τα bits εισέρχονται και εξέρχονται από τις λογικές πύλες. Με την χρήση διατάξεων λογικών πυλών είναι εφικτή η επεξεργασία οποιασδήποτε πληροφορίας για την πραγματοποίηση υπολογισμών [8]. Μία γνωστή λογική πύλη είναι η πύλη NOT.



Εικόνα 3-1. Πύλη NOT

Η πύλη NOT παίρνει ως είσοδο ένα bit και εξάγει ένα bit. Η λειτουργία της είναι η αντιστροφή του bit εισόδου.

Bit εισόδου	*Bit εξόδου*
0	1
1	0

Τα bits, όπως έχει ήδη αναφερθεί, μπορούν να αναπαρασταθούν ως διανύσματα στήλης. Αντίστοιχα, οι λογικές πύλες μπορούν να αναπαρασταθούν με την μορφή πινάκων. Επομένως, η διαδικασία χειρισμού των bits μπορεί να χαρακτηριστεί ως πολλαπλασιασμός πινάκων μεταξύ του πίνακα που αντιπροσωπεύει την λογική πύλη και του πίνακα n × 1 που αντιπροσωπεύει τα bits (διάνυσμα στήλης).

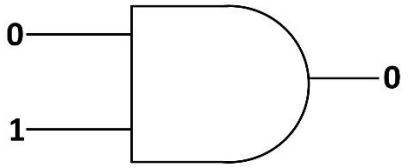
Για παράδειγμα, η πύλη NOT μπορεί να αναπαρασταθεί ως ένας πίνακας 2 × 2.

$$NOT = [0\ 1\ 1\ 0\ ]$$

Έστω ότι το bit εισόδου είναι στην κατάσταση 1. Με την εφαρμογή της λογικής πύλης NOT, η νέα κατάσταση του bit θα είναι η κατάσταση 0, γεγονός που επαληθεύεται με τον πολλαπλασιασμό πινάκων.

$$[0\ 1\ 1\ 0\ ] \times [0\ 1\ ] = [0 * 0 + 1 * 1\ 1 * 0 + 0 * 1\ ] = [1\ 0\ ]$$

Μία άλλη γνωστή λογική πύλη είναι η πύλη AND. Η πύλη AND εκτελεί την λογική πράξη «ΚΑΙ» μεταξύ των εισόδων της. Παίρνει ως είσοδο δύο bits και εξάγει ένα.



Εικόνα 3-2. Πύλη AND

Εξάγει την κατάσταση 1 μόνο αν και τα δύο bits εισόδου βρίσκονται στην κατάσταση 1. Διαφορετικά εξάγει την κατάσταση 0.

Bits εισόδου		***Bit εξόδου***
0	0	0
0	1	0
1	0	0
1	1	1

Η πύλη AND μπορεί να αναπαρασταθεί ως ένας πίνακας 2 × 4, σε αντίθεση με την πύλη NOT, διότι ως είσοδο δέχεται δύο bits αντί για ένα.

$$AND = [1\ 1\ 1\ 0\ \ 0\ 0\ 0\ 1\ \ ]$$

Αξίζει να σημειωθεί ότι τα ονόματα των στηλών αντιστοιχούν στις εισόδους (inputs) και τα ονόματα των γραμμών αντιστοιχούν στις εξόδους (outputs). Για την πύλη AND για παράδειγμα, η πρώτη στήλη του πίνακα αντιστοιχεί στην έξοδο του συστήματος αν η κατάσταση εισόδου είναι η 00, η δεύτερη στήλη του πίνακα αντιστοιχεί στην έξοδο του συστήματος αν η κατάσταση εισόδου είναι η 01 και ούτω καθεξής.

$$00\ 01\ \ 10\ 11\ \ \ 0\ 1\ \ [1\ 1\ 1\ 0\ \ \ 0\ 0\ 0\ 1\ \ \ ]$$

Έστω ότι τα bits εισόδου είναι στην κατάσταση 1 και 0 αντίστοιχα. Η κατάσταση «10» αναπαρίσταται ως ένας πίνακας 4 × 1.

$$10 = [0\ 0\ 1\ 0\ \ ]$$

Υλοποιώντας την λογική πύλη AND με πολλαπλασιασμό πινάκων πάνω στην κατάσταση 10, επαληθεύεται ότι το bit εξόδου είναι η κατάσταση 0.

$$[1\ 1\ 1\ 0\ \ 0\ 0\ 0\ 1\ \ ] \times [0\ 0\ 1\ 0\ \ ] = [1 * 0 + 1 * 0 + 1 * 1 + 0 * 0\ 0 * 0 + 0 * 0 + 0 * 1 + 1 * 0\ ] = [1\ 0\ ]$$

Εναλλακτικά, η κατάσταση εισόδου 10 αντιστοιχεί στην τρίτη στήλη του πίνακα της πύλης AND. Η τρίτη στήλη του πίνακα αντιπροσωπεύει την κατάσταση $0 = [1\ 0\ ]$.

$$00\ 01\ \ 10\ 11\ \ \ 0\ 1\ \ [1\ 1\ 1\ 0\ \ \ 0\ 0\ 0\ 1\ \ \ ]$$

Ο συνδυασμός διαφόρων λογικών πυλών για την πραγματοποίηση συγκεκριμένων υπολογισμών ονομάζεται **κύκλωμα (circuit).**

3.2 Κβαντικές Λογικές Πύλες

Όπως, οι κλασσικές λογικές πύλες αποτελούν τον βασικό τρόπο χειρισμού των bits, έτσι είναι αναγκαία η ύπαρξη ενός αντίστοιχου τρόπου χειρισμού των qubits. Αντίστοιχα με την κλασσικές λογικές πύλες υπάρχουν και οι κβαντικές λογικές πύλες. Δέχονται ως είσοδο ένα ή παραπάνω qubits και εξάγουν ένα ή παραπάνω qubits. Με τον ίδιο τρόπο και οι κβαντικές πύλες μπορούν να αναπαρασταθούν ως πίνακες και η υλοποίηση τους μπορεί να χαρακτηριστεί ως πολλαπλασιασμός πινάκων μεταξύ του πίνακα που αντιπροσωπεύει την κβαντική πύλη και του πίνακα n × 1 που αντιπροσωπεύει τα qubits (διάνυσμα στήλης).

Σε αντίθεση με τις κλασσικές λογικές πύλες, οι κβαντικές λογικές πύλες πρέπει να είναι **αντιστρέψιμες**. Οι αντιστρέψιμες πύλες έχουν αντιστοίχιση ένα προς ένα μεταξύ των καταστάσεων εισόδου και εξόδου. Ισοδύναμα, εφαρμόζοντας την πύλη για δεύτερη φορά στο ίδιο bit το σύστημα επιστρέφεται στην αρχική κατάσταση. Η πύλη NOT για παράδειγμα, αποτελεί μία αντιστρέψιμη πύλη.

Έστω ότι το bit εξόδου από μία πύλη NOT είναι στην κατάσταση 1. Είναι λογικό ότι το bit εισόδου θα πρέπει να βρισκόταν στην κατάσταση 0 προκειμένου το bit εξόδου να είναι στην κατάσταση 1. Εφαρμόζοντας την πύλη NOT για δεύτερη φορά το bit θα βρεθεί στην κατάσταση 0.

$$0\ NOT \rightarrow 1\ NOT \rightarrow 0$$

Αντίστοιχα αν το bit εξόδου είναι στην κατάσταση 0, τότε είναι γνωστό ότι το bit εισόδου θα ήταν στην κατάσταση 1. Εφαρμόζοντας την πύλη NOT για δεύτερη φορά το bit θα βρεθεί στην κατάσταση 1.

Έστω ότι η έξοδος από μία πύλη AND είναι η κατάσταση 0. Η πύλη εξάγει την κατάσταση 0 όταν τα bit εισόδου είναι στις καταστάσεις 00 ή 01 ή 10. Έτσι, από την έξοδο της πύλης AND, δεν είναι εφικτό να καθοριστούν τα bits εισόδου και συνεπώς η πύλη AND δεν θεωρείται αντιστρέψιμη. Ως συνέπεια δεν μπορούν όλες οι κλασσικές λογικές πύλες να χρησιμοποιηθούν στην Κβαντική Υπολογιστική. Στην Κβαντική Υπολογιστική όλες οι λειτουργίες, οι οποίες δεν είναι μετρήσεις, πρέπει να είναι αντιστρέψιμες. Οι κβαντικές πύλες αναπαρίστανται ως μιγαδικοί **μοναδιακοί πίνακες (unitary matrices).**

Ένας μιγαδικός τετραγωνικός πίνακας U είναι μοναδιακός αν ο **ανάστροφος συζυγής (conjugate transpose)** του $U^\dagger$ είναι επίσης ο **αντίστροφός (inverse)** του. Ως αποτέλεσμα:

$$UU^\dagger = U^\dagger U = I$$

όπου I ο μοναδιαίος πίνακας.

Έστω ένας μοναδιακός πίνακας U.

$$U = [a\ c\ b\ d\ ]$$

Ο ανάστροφος συζυγής του δίνεται ως εξής:

$$U^\dagger = \left[\bar{a}\ \bar{b}\ \bar{c}\ \bar{d}\ \right]$$

Συνεπάγεται ότι:

$$UU^{\dagger} = [a\ c\ b\ d\ ] * \left[\bar{a}\ \bar{b}\ \bar{c}\ \bar{d}\ \right] = [1\ 0\ 0\ 1\ ] = I$$

Μια κβαντική πύλη συνεπώς, είναι ένας **τελεστής (operator)** ο οποίος ενεργεί πάνω στα qubits. Τέτοιοι τελεστές αναπαρίστανται ως μοναδιακοί πίνακες. Η εξέλιξη ενός κβαντικού συστήματος δίνεται από έναν μοναδιακό τελεστή ή αλλιώς μετασχηματισμό (unitary operator ή transformation).

Δηλαδή, εάν το U είναι ένας μοναδιακός πίνακας που αντιπροσωπεύει ένα μοναδιακό τελεστή και $|\psi(t)\rangle$ αντιπροσωπεύει την κατάσταση του συστήματος σε μία χρονική στιγμή t, τότε

$$|\psi(t+1)\rangle = U|\psi(t)\rangle$$

θα αντιπροσωπεύει το σύστημα την χρονική στιγμή t+1.

3.2.1 Κβαντικές Πύλες ενός Qubit

Παρακάτω αναφέρονται ορισμένες από τις βασικές κβαντικές πύλες οι οποίες ενεργούν πάνω σε ένα qubit.

3.2.1.1 Bit Flip ή Pauli X gate

Η κβαντική πύλη **Pauli** X ή αλλιώς **Bit Flip** είναι αντίστοιχη της κλασσικής πύλης NOT. Ο πίνακας της πύλης X δίνεται ως εξής:

$$X = (0\ 1\ 1\ 0\ )$$

Η πύλη X αντιστοιχίζει

- την κατάσταση $|0\rangle$ στην κατάσταση $|1\rangle$ και
- την κατάσταση $|1\rangle$ στην κατάσταση $|0\rangle$.

$$X|0\rangle = (0\ 1\ 1\ 0\ )(1\ 0\ ) = (0 * 1 + 1 * 0\ 1 * 1 + 0 * 0\ ) = (0\ 1\ ) = |1\rangle$$

$$X|1\rangle = (0\ 1\ 1\ 0\ )(0\ 1\ ) = (0 * 0 + 1 * 1\ 1 * 0 + 0 * 1\ ) = (0\ 1\ ) = |0\rangle$$

Για μία τυχαία κατάσταση $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ προκύπτει ότι:

$$X|\psi\rangle = (0\ 1\ 1\ 0\ )(\alpha\ \beta\ ) = (0 * \alpha + 1 * \beta\ 1 * \alpha + 0 * \beta\ ) = (\beta\ \alpha\ ) = \beta|0\rangle + \alpha|1\rangle$$

Θα πρέπει ο πίνακας να είναι μοναδιακός (unitary), όπως έχει ήδη αναφερθεί. Δηλαδή:

$$XX^{\dagger} = X^{\dagger}X = I$$

Για τον πίνακα X ισχύει ότι $X = X^{\dagger}$, διότι δεν έχει μιγαδικές τιμές. Επομένως πρέπει να αποδειχθεί ότι:

$$X^2 = I$$

$$X^2 = (0\ 1\ 1\ 0\ )(0\ 1\ 1\ 0\ ) = (0 * 0 + 1 * 1\ 0 * 1 + 1 * 0\ 1 * 0 + 0 * 1\ 1 * 1 + 0 * 0\ ) = (1\ 0\ 0\ 1\ ) = I$$

Άρα ο πίνακας X είναι μοναδιακός.

Το ίδιο θα ισχύει και για όλους τους παρακάτω πίνακες που θα αναφερθούν.

3.2.1.2 Pauli Y gate

Η κβαντική πύλη **Pauli** Y δίνεται ως εξής:

$$Y = (0 \;\; -i \; i \;\; 0)$$

Η πύλη Y αντιστοιχίζει

- την κατάσταση $|0\rangle$ στην κατάσταση $i|1\rangle$ και
- την κατάσταση $|1\rangle$ στην κατάσταση $-i|0\rangle$.

$$Y|0\rangle = (0 \;\; -i \; i \;\; 0)(1 \;\; 0) = (0 * 1 - i * 0 \; i * 1 + 0 * 0) = (0 \; i) = i|1\rangle$$

$$Y|1\rangle = (0 \;\; -i \; i \;\; 0)(0 \;\; 1) = (0 * 0 - i * 1 \; i * 0 + 0 * 1) = (-i \; 0) = -i|0\rangle$$

Για μία τυχαία κατάσταση $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ προκύπτει ότι:

$$Y|\psi\rangle = (0 \;\; -i \; i \;\; 0)(\alpha \; \beta) = (0 * \alpha - i * \beta \; i * \alpha + 0 * \beta) = (-i\beta \; ia) = i(-\beta|0\rangle + \alpha|1\rangle)$$

3.2.1.3 Pauli Z gate

Η κβαντική πύλη **Pauli** Z δίνεται ως εξής:

$$Z = (1 \; 0 \; 0 \;\; -1)$$

Η πύλη Z αντιστοιχίζει

- την κατάσταση $|0\rangle$ στην κατάσταση $|0\rangle$ και
- την κατάσταση $|1\rangle$ στην κατάσταση $-|1\rangle$.

$$Z|0\rangle = (1 \; 0 \; 0 \;\; -1)(1 \;\; 0) = (1 * 1 + 0 * 0 \; 0 * 1 - 1 * 0) = (1 \; 0) = |0\rangle$$

$$Z|1\rangle = (1 \; 0 \; 0 \;\; -1)(0 \;\; 1) = (1 * 0 + 0 * 1 \; 0 * 0 - 1 * 1) = (0 \;\; -1) = -|1\rangle$$

Για μία τυχαία κατάσταση $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ προκύπτει ότι:

$$Z|\psi\rangle = (1 \; 0 \; 0 \;\; -1)(\alpha \; \beta) = (1 * \alpha + 0 * \beta \; 0 * \alpha - 1 * \beta) = (\alpha \;\; -\beta) = \alpha|0\rangle - \beta|1\rangle$$

Οι παραπάνω τρεις πύλες Pauli X, Pauli Y και Pauli Z καλούνται **πίνακες Pauli** (**Pauli matrices**).

3.2.1.4 Hadamard gate

Μία από τις σημαντικότερες κβαντικές πύλες είναι η πύλη Hadamard. Η κβαντική πύλη **Hadamard** δίνεται ως εξής:

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

Εφαρμόζοντας την πύλη Hadamard στην κατάσταση $|0\rangle$ ή $|1\rangle$ δημιουργείται μία ίση υπέρθεση των καταστάσεων $|0\rangle$ και $|1\rangle$. Επομένως μετά από μία μέτρηση, οι δύο καταστάσεις θα έχουν ίδια πιθανότητα εμφάνισης.

Η πύλη H αντιστοιχίζει

- την κατάσταση $|0\rangle$ στην κατάσταση $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = |+\rangle$ και
- την κατάσταση $|1\rangle$ στην κατάσταση $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = |-\rangle$.

$$H|0\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} (1\ 0) = \begin{pmatrix} \frac{1}{\sqrt{2}} * 1 + \frac{1}{\sqrt{2}} * 0 & \frac{1}{\sqrt{2}} * 1 - \frac{1}{\sqrt{2}} * 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} = |+\rangle$$

$$H|1\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} (0\ 1) = \begin{pmatrix} \frac{1}{\sqrt{2}} * 0 + \frac{1}{\sqrt{2}} * 1 & \frac{1}{\sqrt{2}} * 0 - \frac{1}{\sqrt{2}} * 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = |-\rangle$$

Για μία τυχαία κατάσταση $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ προκύπτει ότι:

$$H|\psi\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} (\alpha\ \beta) = \begin{pmatrix} \frac{1}{\sqrt{2}} * \alpha + \frac{1}{\sqrt{2}} * \beta & \frac{1}{\sqrt{2}} * \alpha - \frac{1}{\sqrt{2}} * \beta \end{pmatrix} = \begin{pmatrix} \frac{\alpha+\beta}{\sqrt{2}} & \frac{\alpha-\beta}{\sqrt{2}} \end{pmatrix}$$

3.2.1.5 Identity gate

Η κβαντική πύλη I είναι αντίστοιχη του μοναδιαίου πίνακα I και δίνεται ως εξής:

$$I = (1\ 0\ 0\ 1)$$

Η πύλη I αντιστοιχίζει

- την κατάσταση $|0\rangle$ στην κατάσταση $|0\rangle$ και
- την κατάσταση $|1\rangle$ στην κατάσταση $|1\rangle$.

$$I|0\rangle = (1\ 0\ 0\ 1)(1\ 0) = (1 * 1 + 0 * 0\ 0 * 1 + 1 * 0) = (1\ 0) = |0\rangle$$

$$I|1\rangle = (1\ 0\ 0\ 1)(0\ 1) = (1 * 0 + 0 * 1\ 0 * 0 + 1 * 1) = (0\ 1) = |1\rangle$$

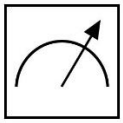
Για μία τυχαία κατάσταση $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ προκύπτει ότι:

$$I|\psi\rangle = (1\ 0\ 0\ 1)(\alpha\ \beta) = (1 * \alpha + 0 * \beta\ 0 * \alpha + 1 * \beta) = (\alpha\ \beta)$$

Μία πύλη I στην ουσία, δεν πραγματοποιεί κάποια αλλαγή στην κατάσταση του qubit.

3.2.1.6 Μέτρηση

Η ίδια η διαδικασία της μέτρησης θεωρείται μία κβαντική πύλη με την διαφορά όμως ότι δεν είναι αντιστρέψιμη. Η μέτρηση συνήθως πραγματοποιείται στο τέλος ενός υπολογισμού προκειμένου να προσδιοριστεί η θέση των qubits. Η κβαντική πύλη της της μέτρησης συμβολίζεται ως ένας δείκτης σε ένα καντράν.

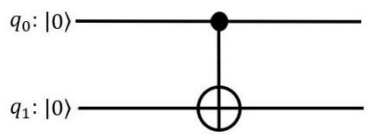


Εικόνα 3-3. Κβαντική Πύλη Μέτρησης

3.2.2 Κβαντικές Πύλες για δύο Qubits

3.2.2.1 Πύλη CNOT (Controlled NOT Gate)

Η κβαντική πύλη CNOT δέχεται ως είσοδο δύο qubits και εξάγει δύο qubits. Το πρώτο qubit εισόδου καλείται **control qubit.** Το δεύτερο qubit εισόδου καλείται **target qubit**. Η πύλη CNOT αφήνει το control qubit αμετάβλητο και αντιστρέφει το target qubit μόνο αν το control qubit είναι στην κατάσταση $|1\rangle$. Αν οι τιμές εισόδου είναι μόνο μία από τις οι δύο βασικές καταστάσεις $|0\rangle$ και $|1\rangle$ και για τα δύο qubits, η έξοδος του target qubit της πύλης CNOT αντιστοιχεί στο αποτέλεσμα της κλασσικής πύλης **XOR (exclusive OR).** Η λογική πράξη της XOR συμβολίζεται ως "$\oplus$". Σχηματικά:



Εικόνα 3-4. Διάγραμμα CNOT

όπου q_0 το control qubit και q_1 το target qubit.

Η πύλη CNOT δίνεται ως εξής:

$$CNOT = (1\ 0\ \ 0\ 0\ \ 0\ 1\ \ 0\ 0\ \ 0\ 0\ \ \ \ 0\ 0\ 0\ 1\ \ 1\ 0\ \ )$$

Η πύλη CNOT αντιστοιχίζει

- την κατάσταση $|00\rangle$ στην κατάσταση $|00\rangle$,
- την κατάσταση $|01\rangle$ στην κατάσταση $|01\rangle$,
- την κατάσταση $|10\rangle$ στην κατάσταση $|11\rangle$,
- την κατάσταση $|11\rangle$ στην κατάσταση $|10\rangle$.

Το target qubit αλλάζει μόνο όταν το πρώτο qubit είναι στην κατάσταση $|1\rangle$, δηλαδή στις καταστάσεις $|10\rangle$ και $|11\rangle$.

Για μία τυχαία κατάσταση $|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$ προκύπτει ότι:

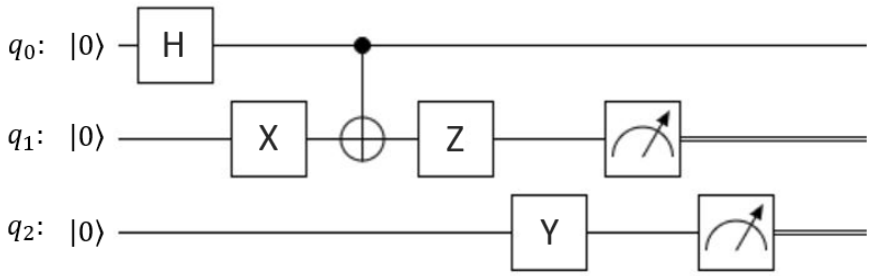
$$CNOT|\psi\rangle = (1\ 0\ \ 0\ 0\ \ 0\ 1\ \ 0\ 0\ \ 0\ 0\ \ \ \ 0\ 0\ 0\ 1\ \ 1\ 0\ \ )(a_{00}\ a_{01}\ a_{10}\ a_{11}\ \ ) = (a_{00}\ a_{01}\ a_{11}\ a_{10}\ \ )$$

3.3 Μοντέλο Προγραμματισμού Κβαντικών Κυκλωμάτων

Η χρήση και η διάταξη διαφόρων κβαντικών πυλών για την πραγματοποίηση συγκεκριμένων υπολογισμών καλείται **κβαντικό κύκλωμα (quantum circuit).** Το μοντέλο προγραμματισμού κβαντικών κυκλωμάτων δίνεται ως εξής:

- Αρχικά, δίνεται πάντα ένα σύνολο από qubits τα οποία βρίσκονται όλα στην κατάσταση $|0\rangle$ και χρησιμοποιούνται ως qubit εισόδου.

- Έπειτα, όπως και με τις κλασσικές πύλες, θεωρείται ότι υπάρχει ένα καλώδιο για κάθε qubit το οποίο μεταφέρει την κβαντική πληροφορία του qubit που του αντιστοιχεί.
- Στην συνέχεια, υλοποιούνται κβαντικές πύλες στα διάφορα qubit, οι οποίες εκτελούν μοναδιακούς μετασχηματισμούς πάνω στα qubit.
- Τέλος, πραγματοποιούνται μία ή περισσότερες μετρήσεις πάνω σε ένα ή περισσότερα qubit για την εξαγωγή των αποτελεσμάτων/συμπερασμάτων.



Εικόνα 3-5. Παράδειγμα Κβαντικού Κυκλώματος με τρία qubits

Συγκεκριμένα, στο παραπάνω κύκλωμα στο πρώτο qubit έχει εφαρμοστεί μία πύλη Hadamard και μία πύλη CNOT, μεταξύ του πρώτου και δεύτερου qubit, με το πρώτο qubit να λειτουργεί ως control qubit. Στο δεύτερο qubit έχει εφαρμοστεί μία πύλη Pauli X, μία πύλη CNOT, μεταξύ του πρώτου και δεύτερου qubit, με το δεύτερο qubit να λειτουργεί ως target qubit, μία πύλη Pauli Z και στο τέλος πραγματοποιείται μία μέτρηση στο δεύτερο qubit. Στο τρίτο qubit έχει εφαρμοστεί μία πύλη Pauli Y και έπειτα πραγματοποιείται μία μέτρηση σε αυτό.

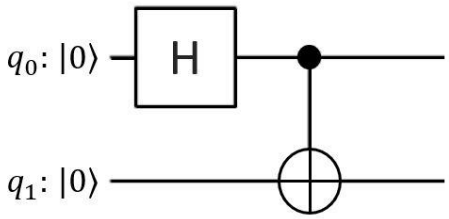
Το παραπάνω κύκλωμα χρησιμοποιήθηκε ως παράδειγμα και δεν υλοποιεί κάτι συγκεκριμένο και συνεπώς δεν έχει κάποια χρησιμότητα. Στο επόμενο κεφάλαιο ακολουθούν μερικά χρήσιμα κβαντικά κυκλώματα και μερικοί βασικοί κβαντικοί αλγόριθμοι.

3.4 Κβαντικοί Αλγόριθμοι

3.4.1 Δημιουργία Καταστάσεων Bell

Η δυνατότητα δημιουργίας καταστάσεων Bell (Bell states), είναι μία βασική τεχνική που χρησιμοποιείται συχνά σε κβαντικούς αλγορίθμους. Για την δημιουργία μίας κατάστασης Bell χρειάζονται δύο qubits, όπως έχει ήδη αναφερθεί.

Ξεκινώντας αρχικά για την κατάσταση Bell $|\Phi^+\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, η οποία αποτελεί και την πιο βασική από τις τέσσερις καταστάσεις Bell. Το κύκλωμα για την δημιουργία της κατάστασης $|\Phi^+\rangle$ δίνεται ως εξής:



Εικόνα 3-6. Κύκλωμα κατάστασης Bell Φ+

Αναλυτικότερα, τα δύο qubits αρχικά βρίσκονται στην κατάσταση $|0\rangle$. Η κατάσταση αυτή του συστήματος μπορεί να εκφραστεί ως εξής:

$$|0\rangle \otimes |0\rangle = |00\rangle$$

Εφαρμόζοντας την πύλη Hadamard στο πρώτο qubit οι νέες καταστάσεις των δύο qubits θα είναι $|q_0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ και $|q_1\rangle = |0\rangle$. Η νέα κατάσταση του συστήματος είναι η εξής:

$$\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes |0\rangle = \left(\frac{1}{\sqrt{2}}|0\rangle \otimes |0\rangle\right) + \left(\frac{1}{\sqrt{2}}|1\rangle \otimes |0\rangle\right) = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle$$

Η κατάσταση αυτή του συστήματος ως διάνυσμα στήλης γράφεται ως εξής: $\begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \end{pmatrix}^T$

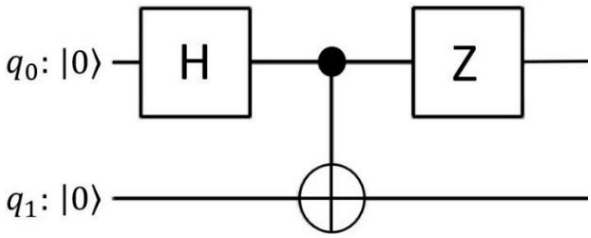
Τέλος, εφαρμόζοντας την πύλη CNOT στα δύο qubits q_0 και q_1, με το q_0 ως control qubit και το q_1 ως target qubit προκύπτει ότι:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{pmatrix}$$

Συνεπώς, η νέα κατάσταση του συστήματος θα είναι η κατάσταση $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle = |\Phi^+\rangle$.

Το κύκλωμα που μόλις περιεγράφηκε είναι κοινό και για τις τέσσερις καταστάσεις Bell με την μόνη διαφορά ότι για τις υπόλοιπες τρεις θα χρειαστεί να εφαρμοστούν επιπλέον μία ή δύο κβαντικές πύλες ακόμα προκειμένου να επιτευχθεί η επιθυμητή κατάσταση.

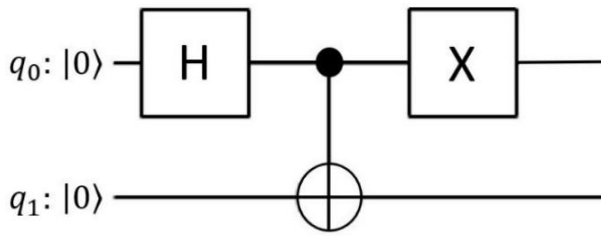
Για την κατάσταση $|\Phi^-\rangle = \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle$, το κύκλωμα που χρειάζεται είναι το εξής:



Εικόνα 3-7. Κύκλωμα κατάστασης Bell Φ-

Με την προσθήκη της πύλης Pauli Z, η κατάσταση του συστήματος είναι η $|\Phi^-\rangle = \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle$.

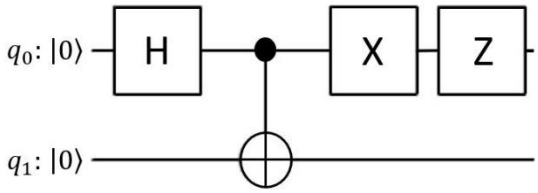
Για την κατάσταση $|\Psi^+\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle$, το κύκλωμα που χρειάζεται είναι το εξής:



Εικόνα 3-8. Κύκλωμα κατάστασης Bell Ψ+

Με την προσθήκη της πύλης Pauli X, η κατάσταση του συστήματος είναι η $|\Psi^+\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle$.

Για την κατάσταση $|\Psi^-\rangle = \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle$, το κύκλωμα που χρειάζεται είναι το εξής:



Εικόνα 3-9. Κύκλωμα κατάστασης Bell Ψ-

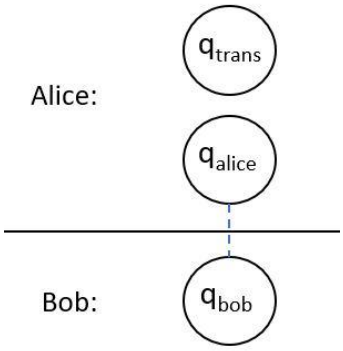
Με την προσθήκη των πυλών Pauli X και Pauli Z, η κατάσταση του συστήματος είναι η $|\Psi^-\rangle = \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle$.

3.4.2 Κβαντική Τηλεμεταφορά

Έστω μία τυχαία άγνωστη κβαντική κατάσταση $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, όπου τα πλάτη πιθανότητας α, β δεν είναι γνωστά. Το **Θεώρημα Μη Κλωνοποίησης (No Cloning Theorem)** δηλώνει ότι είναι αδύνατο να δημιουργηθεί ένα πανομοιότυπο αντίγραφο μιας αυθαίρετης άγνωστης κβαντικής κατάστασης [6]. Στην Κβαντική Μηχανική όμως είναι εφικτή η μετάδοση μίας τυχαίας άγνωστης κβαντικής κατάστασης από ένα σημείο σε ένα άλλο. Η διαδικασία αυτή καλείται **Κβαντική Τηλεμεταφορά (Quantum Teleportation).** Η κβαντική τηλεμεταφορά εκτελείται με τον εξής τρόπο:

Έστω δύο φίλοι, η Alice και ο Bob. Η Alice έχει ένα qubit σε μία τυχαία κβαντική κατάσταση $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, το οποίο θέλει να μεταβιβάσει στον Bob. Αν πραγματοποιήσει μία μέτρηση στο qubit της, η κβαντική κατάσταση θα καταρρεύσει σε μία από τις δύο βασικές καταστάσεις $|0\rangle$ ή $|1\rangle$ και συνεπώς η κβαντική πληροφορία του qubit της θα χαθεί. Προκειμένου να επιτευχθεί η μετάδοση του qubit της στον Bob θα χρειαστεί η Alice και ο Bob επιπλέον να μοιράζονται ένα διαπλεγμένο ζεύγος από

qubit, το ένα qubit εκ των οποίων θα ανήκει στην Alice και το άλλο στον Bob, όπως παρουσιάζεται στην παρακάτω εικόνα.

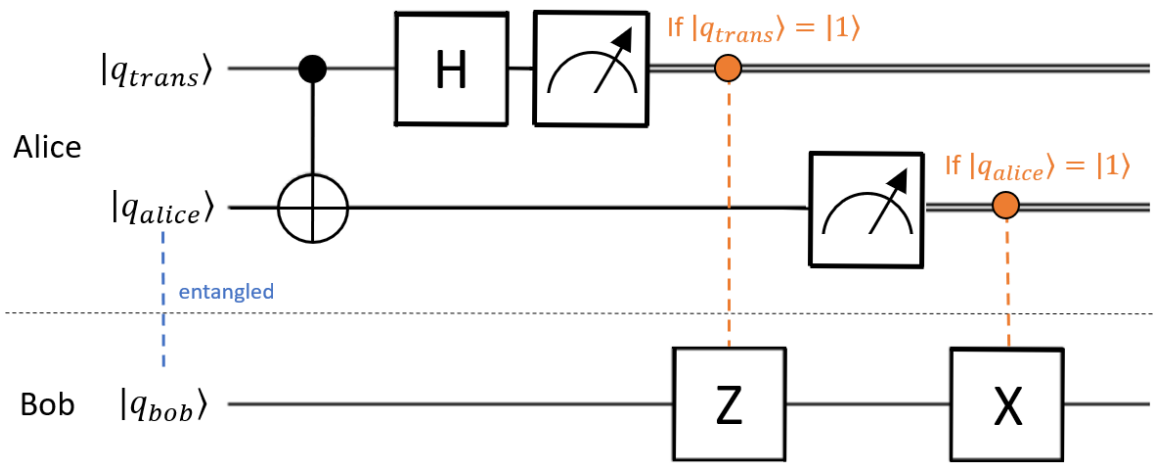


Εικόνα 3-10. Qubit Alice και Bob

όπου q_{trans} το qubit που θέλει να μεταδώσει η Alice, q_{alice} και q_{bob} τα διαπλεγμένα qubit της Alice του Bob αντίστοιχα.

Σύμφωνα με το πρωτόκολλο Κβαντικής Τηλεμεταφοράς, χρειάζονται τέσσερα βασικά βήματα:

1. η Alice αρχικά θα πρέπει να εφαρμόσει μία πύλη CNOT μεταξύ των δύο qubit της, όπου το q_{trans} θα είναι το control qubit και το q_{alice} θα είναι το target qubit.
2. η Alice εφαρμόζει μία πύλη Hadamard στο q_{trans} της.
3. η Alice πραγματοποιεί μία μέτρηση στο $q_{trans.}$
    a. Αν η μέτρηση δείξει την κατάσταση $|0\rangle$, τότε ο Bob δεν πραγματοποιεί κάποια αλλαγή στο qubit του.
    b. Αν η μέτρηση δείξει την κατάσταση $|1\rangle$, τότε ο Bob εφαρμόζει μία πύλη Pauli Z στο q_{bob}.
4. η Alice πραγματοποιεί μία μέτρηση στο q_{alice}.
    a. Αν η μέτρηση δείξει την κατάσταση $|0\rangle$, τότε ο Bob δεν πραγματοποιεί κάποια αλλαγή στο qubit του.
    b. Αν η μέτρηση δείξει την κατάσταση $|1\rangle$, τότε ο Bob εφαρμόζει μία πύλη Pauli X στο q_{bob}.



Εικόνα 3-11. Κύκλωμα Κβαντικής Τηλεμεταφοράς

Αρχικά, το qubit που θέλει να μεταδώσει η Alice στον Bob βρίσκεται στην τυχαία κβαντική κατάσταση $|q_{trans}\rangle = \alpha|0\rangle + \beta|1\rangle$ και το δεύτερο της qubit, q_{alice}, είναι διαπλεγμένο με το qubit του Bob, q_{bob}, και σχηματίζουν την κατάσταση $|q_{alice}\rangle \otimes |q_{bob}\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle = |\Phi^{+}\rangle$. Επομένως, το κβαντικό σύστημα συνολικά βρίσκεται στην εξής κατάσταση:

$$|q_{trans}\rangle \otimes |q_{alice}\rangle \otimes |q_{bob}\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes \left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle\right) =$$

$$\frac{\alpha}{\sqrt{2}}|000\rangle + \frac{\alpha}{\sqrt{2}}|011\rangle + \frac{\beta}{\sqrt{2}}|100\rangle + \frac{\beta}{\sqrt{2}}|111\rangle$$

Υπενθυμίζοντας, ο τρόπος με τον οποίο παρουσιάζονται οι πιθανές καταστάσεις για τα 3 qubit είναι της μορφής $|\alpha\beta\gamma\rangle$ όπου:

- το α αντιπροσωπεύει το q_{trans}
- το β αντιπροσωπεύει το q_{alice}
- το γ αντιπροσωπεύει το q_{bob}

Για παράδειγμα, η κατάσταση $|011\rangle$ δηλώνει ότι το $|q_{trans}\rangle = |0\rangle$, το $|q_{alice}\rangle = |1\rangle$ και το $|q_{bob}\rangle = |1\rangle$.

Κατά το πρώτο βήμα, εφαρμόζεται η πύλη CNOT μεταξύ των δύο qubit της Alice. Επομένως, όπου $|q_{trans}\rangle = |1\rangle$ (control qubit), η κατάσταση $|q_{alice}\rangle$ (target qubit) θα αντιστρέφεται. Για παράδειγμα, η κατάσταση $|100\rangle$, μετά την πύλη CNOT θα αλλάξει στην $|110\rangle$. Η κατάσταση του συστήματος μετά την CNOT θα είναι η εξής:

$$\frac{\alpha}{\sqrt{2}}|000\rangle + \frac{\alpha}{\sqrt{2}}|011\rangle + \frac{\beta}{\sqrt{2}}|110\rangle + \frac{\beta}{\sqrt{2}}|101\rangle$$

Ισοδύναμα, για την ευκολότερη κατανόηση του επόμενου βήματος, ανασυγκροτώντας τις τριάδες $|\alpha\beta\gamma\rangle$ ως $|\alpha\rangle \otimes |\beta\gamma\rangle$, η παραπάνω κατάσταση του συστήματος μπορεί να γραφτεί και ως εξής:

$$\frac{\alpha}{\sqrt{2}}|0\rangle \otimes (|00\rangle + |11\rangle) + \frac{\beta}{\sqrt{2}}|1\rangle \otimes (|10\rangle + |01\rangle)$$

Κατά το δεύτερο βήμα εφαρμόζεται η πύλη Hadamard στο q_{trans} της Alice. Ως αποτέλεσμα,

- όπου το $|q_{trans}\rangle = |0\rangle$, θα αλλάζει στην $|q_{trans}\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ και
- όπου $|q_{trans}\rangle = |1\rangle$, θα αλλάζει στην $|q_{trans}\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$.

Η κατάσταση του συστήματος σε αυτό το σημείο θα είναι η εξής:

$$\frac{\alpha}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes (|00\rangle + |11\rangle) + \frac{\beta}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) \otimes (|10\rangle + |01\rangle) =$$

$$\frac{\alpha}{2}(|0\rangle + |1\rangle) \otimes (|00\rangle + |11\rangle) + \frac{\beta}{2}(|0\rangle - |1\rangle) \otimes (|10\rangle + |01\rangle) =$$

$$\frac{\alpha}{2}(|000\rangle + |011\rangle + |100\rangle + |111\rangle) + \frac{\beta}{2}(|010\rangle + |001\rangle - |110\rangle - |101\rangle)$$

Όπως και προηγουμένως, ανασυγκροτώντας τις τριάδες $|\alpha\beta\gamma\rangle$ ως $|\alpha\beta\rangle \otimes |\gamma\rangle$, η παραπάνω κατάσταση του συστήματος μπορεί να γραφτεί και ως εξής:

$$\frac{1}{2}(|00\rangle(\alpha|0\rangle + \beta|1\rangle)) + \frac{1}{2}(|01\rangle(\beta|0\rangle + \alpha|1\rangle)) + \frac{1}{2}(|10\rangle(\alpha|0\rangle - \beta|1\rangle)) + \frac{1}{2}(|11\rangle(-\beta|0\rangle + \alpha|1\rangle))$$

Τέλος, μετά από τις δύο μετρήσεις που πραγματοποιεί η Alice στα δύο qubit της, q_{trans} και q_{alice}, υπάρχουν τέσσερα πιθανά ενδεχόμενα.

- Αν $|q_{trans}\rangle = |0\rangle$ και $|q_{alice}\rangle = |0\rangle$ ($|\alpha\beta\rangle = |00\rangle$), τότε $|q_{bob}\rangle = \alpha|0\rangle + \beta|1\rangle$
- Αν $|q_{trans}\rangle = |0\rangle$ και $|q_{alice}\rangle = |1\rangle$ ($|\alpha\beta\rangle = |01\rangle$), τότε $|q_{bob}\rangle = \beta|0\rangle + \alpha|1\rangle$
- Αν $|q_{trans}\rangle = |1\rangle$ και $|q_{alice}\rangle = |0\rangle$ ($|\alpha\beta\rangle = |10\rangle$), τότε $|q_{bob}\rangle = \alpha|0\rangle - \beta|1\rangle$
- Αν $|q_{trans}\rangle = |1\rangle$ και $|q_{alice}\rangle = |1\rangle$ ($|\alpha\beta\rangle = |11\rangle$), τότε $|q_{bob}\rangle = -\beta|0\rangle + \alpha|1\rangle$

Αν η Alice μετρήσει $|q_{trans}\rangle = |1\rangle$, ενημερώνει τον Bob ο οποίος με την σειρά του εφαρμόζει μία πύλη Pauli Z στο qubit του προκειμένου να το φέρει στην επιθυμητή κατάσταση. Επίσης, αν η Alice μετρήσει $|q_{alice}\rangle = |1\rangle$, ενημερώνει τον Bob και ο Bob με την σειρά του εφαρμόζει μία πύλη Pauli X στο qubit του προκειμένου να το φέρει στην επιθυμητή κατάσταση.

Για παράδειγμα, έστω το σενάριο όπου η Alice μετράει $|q_{trans}\rangle = |0\rangle$ και $|q_{alice}\rangle = |1\rangle$, το οποίο σημαίνει ότι το qubit του Bob βρίσκεται στην κατάσταση $\beta|0\rangle + \alpha|1\rangle$. Από την στιγμή που το $|q_{alice}\rangle = |1\rangle$, ο Bob θα εφαρμόσει μία πύλη Pauli X στο qubit του και η νέα κατάσταση του θα είναι $|q_{bob}\rangle = \alpha|0\rangle + \beta|1\rangle$, η οποία είναι και η επιθυμητή κατάσταση. Επομένως, το qubit της Alice έχει μεταδοθεί επιτυχώς στο qubit του Bob.

Αξίζει να σημειωθεί ότι κατά το τέλος του αλγορίθμου, το qubit q_{trans} που επιθυμούσε να μεταδώσει η Alice δεν βρίσκεται πλέον στην κατάσταση που βρισκόταν αρχικά και βρίσκεται σε μία από τις δύο βασικές καταστάσεις $|0\rangle$ ή $|1\rangle$. Προκειμένου δηλαδή να μεταδοθεί η τυχαία κατάσταση $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ από το q_{trans} της Alice στο q_{bob} του Bob, χρειάζεται να «καταστραφεί» η κατάσταση του q_{trans}. Το q_{bob} μόνο βρίσκεται πλέον στην τυχαία κατάσταση $\alpha|0\rangle + \beta|1\rangle$.

3.4.3 Αλγόριθμός Bernstein-Vazirani

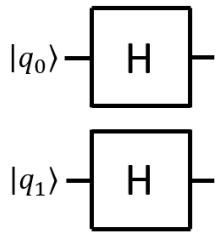
Το 1993 οι **Ethan Bernstein** και **Umesh Vazirani** ήταν από τους πρώτους που σχεδίασαν ένα πρόβλημα το οποίο μπορεί να επιλυθεί γρηγορότερα σε έναν κβαντικό υπολογιστή απ' ότι σε έναν κλασσικό

υπολογιστή. Αν και το ίδιο το πρόβλημα έχει μικρή πρακτική αξία, σκοπός του ήταν να αποδείξει ότι υπάρχουν κβαντικοί αλγόριθμοι οι οποίοι μπορούν να επιλύσουν ορισμένα προβλήματα ταχύτερα από οποιοδήποτε γνωστό κλασσικό αλγόριθμο.

Για την κατανόηση του προβλήματος θα χρειαστεί να αναφερθεί συνοπτικά μία ειδική περίπτωση της τεχνικής που ονομάζεται Fourier Sampling. Η τεχνική αυτή αποτελεί βασικό στοιχείο της κβαντικής μηχανικής και χρησιμοποιείται συχνά σε κβαντικούς υπολογισμούς.

3.4.3.1 Fourier Sampling

Έστω ένα κύκλωμα με δύο qubits σε μία από τις δύο βασικές καταστάσεις, όπου και για τα δύο qubits εφαρμόζεται μία πύλη Hadamard.



Εικόνα 3-12. Κύκλωμα με δύο πύλες Hadamard

Υπάρχουν συνεπώς τέσσερα πιθανά ενδεχόμενα, $|q_0q_1\rangle = |00\rangle$, $|q_0q_1\rangle = |01\rangle$, $|q_0q_1\rangle = |10\rangle$, $|q_0q_1\rangle = |11\rangle$.

Εξετάζοντας την κατάσταση του συστήματος μετά την εφαρμογή των δύο πυλών για κάθε ενδεχόμενο:

$$|00\rangle \rightarrow \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$$

$$|01\rangle \rightarrow \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle$$

$$|10\rangle \rightarrow \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle - \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle$$

$$|11\rangle \rightarrow \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle - \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$$

Σκοπός εδώ είναι να προσδιοριστεί σε ποια σημεία της υπέρθεσης προκύπτει αρνητικό πρόσημο.

$$|00\rangle \rightarrow \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$$

$$|01\rangle \rightarrow \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle$$

$$|10\rangle \rightarrow \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle - \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle$$

$$|11\rangle \rightarrow \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle - \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$$

Γίνεται αντιληπτό ότι, αρνητικό πρόσημο προκύπτει όταν ο αριθμός των άσσων σε μία κατάσταση είναι περιττός. Για παράδειγμα, στο τέταρτο ενδεχόμενο όπου τα qubits εισόδου βρίσκονταν στην κατάσταση $|11\rangle$ προέκυψε η κατάσταση « $\frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle - \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$ ». Αρνητικά πρόσημα προέκυψαν στις

καταστάσεις $|01\rangle$ και $|10\rangle$, διότι η κάθε μία έχει από έναν άσσο, ενώ η κατάσταση $|11\rangle$ έχει θετικό πρόσημο διότι ο αριθμός των άσσων είναι ίσος με δύο (άρτιος αριθμός).

Γενικεύοντας αυτή την αντίληψη, εφαρμόζοντας πύλες Hadamard σε μία κατάσταση $|\alpha\rangle$ των n-qubits (αρχικοποιημένα είτε στην κατάσταση $|0\rangle$ είτε στην κατάσταση $|1\rangle$) προκύπτει το εξής:

$$H^{\otimes n}|\alpha\rangle = \sum_{x \in \{0,1\}^n} \frac{1}{\sqrt{2^n}}(-1)^{\alpha \cdot x}|x\rangle$$

όπου $\alpha \cdot x = (\alpha_1 * x_1 + \alpha_2 * x_2 + \ldots + \alpha_n * x_n)$.

Επαληθεύοντας τον παραπάνω ισχυρισμό χρησιμοποιώντας το προηγούμενο παράδειγμα με τα 2 qubits στην κατάσταση $|11\rangle = |\alpha_1\alpha_2\rangle = |\alpha\rangle$:

$$H^{\otimes n}|\alpha\rangle = H^{\otimes 2}|11\rangle = \sum_{x \in \{0,1\}^2} \frac{1}{\sqrt{2^2}}(-1)^{\alpha \cdot x}|x\rangle =$$

$$\tfrac{1}{2}|00\rangle(-1)^{(1*0+1*0)} + \tfrac{1}{2}|01\rangle(-1)^{(1*0+1*1)} + \tfrac{1}{2}|10\rangle(-1)^{(1*1+1*0)} + \tfrac{1}{2}|11\rangle(-1)^{(1*1+1*1)} =$$

$$\tfrac{1}{2}|00\rangle - \tfrac{1}{2}|01\rangle - \tfrac{1}{2}|10\rangle + \tfrac{1}{2}|11\rangle$$

Αξίζει να αναφερθεί και η απλή περίπτωση κατά την οποία όλα τα qubits είναι αρχικοποιημένα στην κατάσταση $|0\rangle$. Σε αυτή την περίπτωση, η νέα κατάσταση που προκύπτει είναι της μορφής:

$$H^{\otimes n}|00\ldots0\rangle = \sum_{x \in \{0,1\}^n} \frac{1}{\sqrt{2^n}}|x\rangle$$

Επιστρέφοντας στο πρόβλημα που έθεσαν οι Bernstein και Vazirani.

Έστω μία συνάρτηση $f : \{0,1\}^n \rightarrow \{0,1\}$, (input: n-bits, output: 1 bit)

όπου $f(x) = u \cdot x \ (mod 2) = (u_1 * x_1 + u_2 * x_2 + \ldots + u_n * x_n)(mod 2)$ για μία άγνωστη συμβολοσειρά $u \in \{0,1\}^n$.

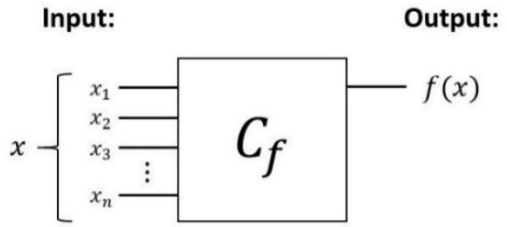
Σκοπός είναι να προσδιοριστεί η άγνωστη συμβολοσειρά u.

Η συνάρτηση f δίνεται ως «μαύρο κουτί» (black box), δηλαδή ως ένα σύστημα το οποίο μπορεί να προσδιοριστεί μόνο από την άποψη των εισόδων (inputs) και των εξόδων (outputs) του, χωρίς να είναι γνωστή η εσωτερική λειτουργία του. Ο μόνος τρόπος για τον προσδιορισμό της συμβολοσειράς u είναι με τροφοδότηση διαφορετικών inputs στο σύστημα και την συλλογή διαφορετικών outputs.

Για παράδειγμα, έστω ότι $n = 4$ και $u = 1011$. Τροφοδοτώντας το σύστημα με input $x = 1001$ προκύπτει το εξής output:

$$f(1001) = u \cdot 1001(mod 2) = 1011 \cdot 1001\,(mod 2) = (1 * 1 + 0 * 0 + 1 * 0 + 1 * 1)\,(mod 2) = 0$$

Ένα κλασσικό κύκλωμα που υλοποιεί την παραπάνω συνάρτηση θα είναι της εξής μορφής:



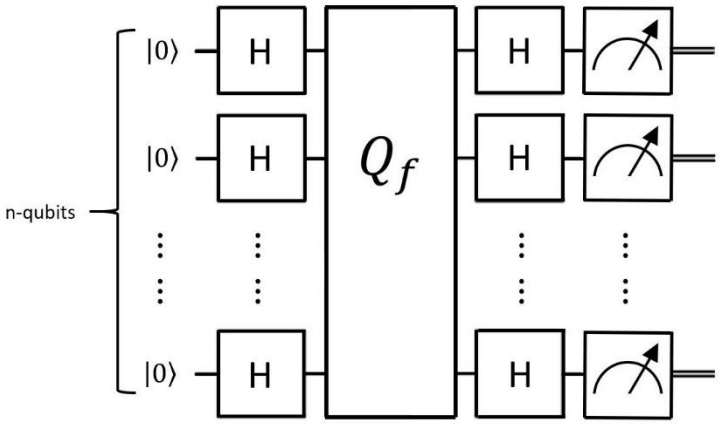
Εικόνα 3-13. Κλασσικό κύκλωμα Bernstein-Vazirani

Η κλασσική προσέγγιση για τον προσδιορισμό της συμβολοσειράς u είναι η τροφοδότηση του συστήματος με τα εξής inputs:

- $x = 100\ldots0 \Rightarrow f(x) = u_1$
- $x = 010\ldots0 \Rightarrow f(x) = u_2$
- $\vdots$
- $x = 000\ldots1 \Rightarrow f(x) = u_n$

Κάθε ένα από τα παραπάνω inputs δίνει ως output το i-οστο στοιχείο της συμβολοσειράς u. Έπειτα από 'n' inputs, έχουν προσδιοριστεί όλα τα $u_1, u_2, \ldots, u_n$ και συνεπώς είναι εφικτή η επανάκτηση της συμβολοσειράς u. Η συγκεκριμένη προσέγγιση καλείται **brute force** διότι για τον προσδιορισμό κάθε bit χρειάζεται ξεχωριστό input.

Με την κβαντική προσέγγιση, σύμφωνα με τον αλγόριθμο Bernstein-Vazirani, η επανάκτηση της συμβολοσειράς u μπορεί να γίνει με την τροφοδότηση μόνο ενός input. Το κύκλωμα που περιγράφει τον αλγόριθμο δίνεται ως εξής:



Εικόνα 3-14. Κβαντικό Κύκλωμα Bernstein-Vazirani

Ο αλγόριθμος αποτελείται από τέσσερα βήματα.

1. Εφαρμογή της πύλης Hadamard σε κάθε qubit.
2. Εφαρμογή του μαύρου κουτιού Q_f, το οποίο υλοποιεί την συνάρτηση f.
3. Εφαρμογή της πύλης Hadamard σε κάθε qubit.

4. Μέτρηση κάθε qubit.

Αρχικά, όλα τα qubits βρίσκονται στην κατάσταση $|0\rangle$. Μετά το πρώτο βήμα του αλγορίθμου, αφού έχουν υλοποιηθεί όλες οι πύλες Hadamard, η κατάσταση του συστήματος βρίσκεται σε μία ίση υπέρθεση όλων των πιθανών καταστάσεων.

$$H^{\otimes n}|00\ldots0\rangle = \sum_{x \in \{0,1\}^n} \frac{1}{\sqrt{2^n}}|x\rangle$$

Κατά το δεύτερο βήμα του αλγορίθμου, υλοποιείται ο μετασχηματισμός Q_f. Ο μοναδιακός μετασχηματισμός Q_f μπορεί να αποσυντεθεί σε n-κβαντικές πύλες, όπου η κάθε μία δρα στο qubit που της αντιστοιχεί. Επομένως ο μετασχηματισμός Q_f μπορεί να γραφτεί ως τανυστικό γινόμενο μεταξύ αυτών των κβαντικών πυλών.

$$Q_f = Q_1 \otimes Q_2 \otimes \ldots Q_n$$

Κάθε πύλη Q_i υλοποιείται κατά τον εξής τρόπο [9]:

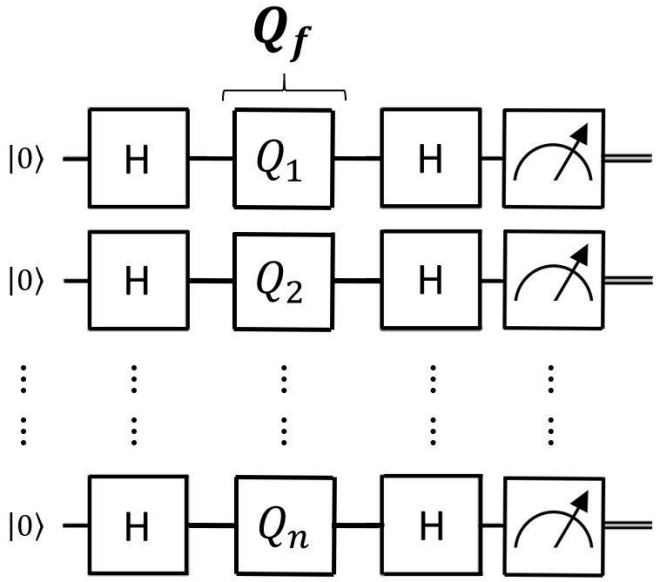
$$Q_i = \{I \text{ , αν } u_i = 0 \; Z \text{ , αν } u_i = 1$$

Δηλαδή, αν το i-οστο στοιχείο της συμβολοσειράς u είναι ίσο με 0 τότε το Q_i δεν πραγματοποιεί κάποια αλλαγή στο i-οστο qubit (ισοδύναμα είναι σαν να εφαρμόζεται ο μοναδιαίος πίνακας I στο qubit).

$$|q_i\rangle \; u_i = 0 \rightarrow \; Q_i|q_i\rangle = I|q_i\rangle = (1\;0\;0\;1\,)\,|q_i\rangle = |q_i\rangle$$

Αν το i-οστο στοιχείο της συμβολοσειράς u είναι ίσο με 1 τότε το Q_i εφαρμόζει την πύλη Pauli-Z στο i-οστο qubit, Q_i = Z .

$$|q_i\rangle \; u_i = 1 \rightarrow \; Q_i|q_i\rangle = Z|q_i\rangle = (1\;0\;0\;-1\,)\,|q_i\rangle$$



Εικόνα 3-15. Κβαντικό Κύκλωμα Bernstein-Vazirani (με πύλες Q_i)

Ως αποτέλεσμα των παραπάνω, η επίδραση που θα έχει ο μετασχηματισμός Q_f σε μία πιθανή κατάσταση του συστήματος θα είναι η εξής:

$$|x\rangle\ Q_f \rightarrow\ (-1)^{f(x)}|x\rangle = (-1)^{u\cdot x}|x\rangle$$

Επομένως, η νέα κατάσταση του συστήματος, μετά το δεύτερο βήμα του αλγορίθμου, γράφεται ως:

$$\sum_{x\,\epsilon\,\{0,1\}^n} \frac{1}{\sqrt{2^n}}(-1)^{f(x)}|x\rangle = \sum_{x\,\epsilon\,\{0,1\}^n} \frac{1}{\sqrt{2^n}}(-1)^{u\cdot x}|x\rangle$$

Κατά το τρίτο βήμα του αλγορίθμου υλοποιούνται για άλλη μια φορά οι πύλες Hadamard σε κάθε ένα από τα n-qubits.

Έχει ήδη αναφερθεί, στο κεφάλαιο *Fourier Sampling*, η επίδραση που έχει ο μετασχηματισμός $\mathrm{H}^{\otimes n}$ σε μία τυχαία κατάσταση $|\alpha\rangle$ των n-qubits (αρχικοποιημένα είτε στην κατάσταση $|0\rangle$ είτε στην κατάσταση $|1\rangle$).

$$\mathrm{H}^{\otimes n}|\alpha\rangle = \sum_{x\,\epsilon\,\{0,1\}^n} \frac{1}{\sqrt{2^n}}(-1)^{\alpha\cdot x}|x\rangle$$

Είναι επίσης γνωστό ότι αν εφαρμοστεί μία πύλη Hadamard δύο φορές σε ένα qubit, η κατάσταση του qubit επανέρχεται στην αρχική του κατάσταση (στην κατάσταση δηλαδή που είχε πριν υλοποιηθεί η πρώτη πύλη Hadamard). Αυτό συμβαίνει διότι η πύλη Hadamard είναι αντιστρέψιμη (ο μετασχηματισμός Hadamard είναι μοναδιακός). Συνδυάζοντας τα δύο παραπάνω προκύπτει ότι αν εφαρμοστεί ο μετασχηματισμός $\mathrm{H}^{\otimes n}$ στην κατάσταση $\mathrm{H}^{\otimes n}|\alpha\rangle$, η κατάσταση του συστήματος θα είναι η κατάσταση $|\alpha\rangle$.

$$\mathrm{H}^{\otimes n}|\alpha\rangle\ \mathrm{H}^{\otimes n} \rightarrow |\alpha\rangle$$

Στην ουσία, οι Bernstein και Vazirani σχεδίασαν το συγκεκριμένο πρόβλημα βασισμένοι πάνω σε αυτήν ακριβώς την ιδιότητα. Συνεπώς, η κατάσταση του συστήματος μετά το τρίτο βήμα του αλγορίθμου:

$$\sum_{x\,\epsilon\,\{0,1\}^n} \frac{1}{\sqrt{2^n}}(-1)^{u\cdot x}|x\rangle\ \mathrm{H}^{\otimes n} \rightarrow |u\rangle$$

Η κατάσταση δηλαδή του συστήματος μετά το τρίτο βήμα θα είναι η συμβολοσειρά u. Το μόνο που μένει είναι να μετρηθούν όλα τα qubits. Το αποτέλεσμα των μετρήσεων θα δώσει την κατάσταση $|u\rangle = |u_1 u_2 \ldots u_n\rangle$. Κάθε i-οστο qubit θα βρίσκεται στην κατάσταση που συμβολίζει το i-οστο στοιχείο της συμβολοσειράς u.

Χρησιμοποιώντας, τον αλγόριθμο των Bernstein και Vazirani η συμβολοσειρά u μπορεί να ανακτηθεί μόνο με ένα τρέξιμο του αλγορίθμου, σε αντίθεση με τον κλασσικό αλγόριθμο ο οποίος χρειάζεται να τροφοδοτήσει το σύστημα με n διαφορετικά inputs.

3.5 Κβαντικοί Υπολογιστές

Κβαντικός υπολογιστής θεωρείται μία υπολογιστική συσκευή που εκμεταλλεύεται χαρακτηριστικές ιδιότητες της Κβαντικής Μηχανικής, όπως την **υπέρθεση** και την **διεμπλοκή**, με σκοπό την επεξεργασία δεδομένων και την εκτέλεση υπολογισμών. Σε έναν κλασσικό υπολογιστή, οι πληροφορίες αποθηκεύονται ως bits, ενώ σε έναν κβαντικό υπολογιστή, οι πληροφορίες αποθηκεύονται ως qubits. Αντίστοιχα με τον τρόπο που ένας κλασσικός υπολογιστής είναι κατασκευασμένος από ένα ηλεκτρικό κύκλωμα το οποίο περιέχει καλώδια και λογικές πύλες, ένας κβαντικός υπολογιστής είναι κατασκευασμένος από ένα κβαντικό κύκλωμα το οποίο περιέχει καλώδια και στοιχειώδεις κβαντικές πύλες οι οποίες μεταφέρουν και διαχειρίζονται την κβαντική πληροφορία.

Οι κβαντικοί υπολογιστές είναι εξαιρετικά ευαίσθητοι σε διαταραχές, θόρυβο και άλλες περιβαλλοντικές επιπτώσεις, οι οποίες επηρεάζουν ή καταστρέφουν την κβαντική τους κατάσταση. Το φαινόμενο αυτό αποκαλείται **Κβαντική Αποσυνοχή (Quantum Decoherence)** και είναι ο βασικός λόγος που μέχρι και σήμερα δεν έχουν κατασκευαστεί μεγάλης κλίμακας κβαντικοί υπολογιστές. Προκειμένου να επιτευχθούν ακριβή αποτελέσματα, χρησιμοποιούνται συγκεκριμένες μονάδες ψύξης ικανές να διατηρήσουν την θερμοκρασία του συστήματος λίγο πιο πάνω από το **απόλυτο μηδέν (absolute zero)**. Καθώς ο αριθμός των qubit σε έναν κβαντικό υπολογιστή αυξάνεται, η αποτελεσματικότητα αυτών των μονάδων ψύξης μειώνεται σημαντικά. Οι επιστήμονες μπορούν να κρατήσουν την κβαντική κατάσταση μόνο για ένα μικρό χρονικό διάστημα (περίπου 90 microseconds, μέχρι και την παρούσα εργασία) προτού χάσει την αξιοπιστία της, γεγονός που περιορίζει αρκετά τον αριθμό των υπολογισμών που μπορούν να πραγματοποιηθούν πριν καταρρεύσει η πληροφορία.

Επομένως, ακόμη και με τα απαραίτητα μέτρα, ο θόρυβος δεν μπορεί ακόμα να αποφευχθεί, οδηγώντας σε απροσδόκητα αποτελέσματα των υπολογισμών. Τα σφάλματα είναι εγγενή στα κβαντικά συστήματα σήμερα και η σταθερή διόρθωση σφαλμάτων μαζί με την αύξηση των qubit είναι κρίσιμα για την πραγματοποίηση της Κβαντικής Υπεροχής.

Μέχρι και την παρούσα εργασία, έχουν κατασκευαστεί κβαντικοί υπολογιστές από εταιρίες όπως η Intel, η IBM, η Google κ.α. Συγκεκριμένα:

- η Intel με τον κβαντικό υπολογιστή 'Tangle Lake' με 49 qubits
- η IBM με τον κβαντικό υπολογιστή 'IBM Q System One' με 50 qubits
- η Google με τον κβαντικό υπολογιστή 'Bristlecone' με 72 qubits

Μεγάλης κλίμακας κβαντικοί υπολογιστές αναμένεται να λύσουν προβλήματα πολύ ταχύτερα από τους κλασσικούς υπολογιστές, χρησιμοποιώντας τους καλύτερους μέχρι τώρα γνωστούς αλγόριθμους όπως για παράδειγμα ο αλγόριθμος του Shor για την παραγοντοποίηση μεγάλων αριθμών.

3.6 Κβαντικές Γλώσσες Προγραμματισμού και Προσομοίωση Κβαντικών Κυκλωμάτων

Τα τελευταία χρόνια, έχουν υλοποιηθεί **κβαντικές γλώσσες προγραμματισμού** καθώς και **κιτ ανάπτυξης κβαντικού λογισμικού (quantum software development kits)** ικανά να προσομοιώσουν κβαντικά κυκλώματα αλλά ακόμα και να υλοποιήσουν κβαντικούς αλγόριθμους σε πραγματικούς κβαντικούς υπολογιστές. Ορισμένα παραδείγματα γλωσσών προγραμματισμού:

- **QCL** (Quantum Computation Language), μία από τις πρώτες γλώσσες που αναπτύχθηκαν της οποίας η σύνταξη είναι παρόμοια με την γλώσσα προγραμματισμού C.

- **Quipper**, εξίσου μία από τις πρώτες γλώσσες που αναπτύχθηκαν η οποία βασίζεται στην γλώσσα προγραμματισμού Haskell.
- **Q Language,** υλοποιήθηκε ως επέκταση της γλώσσας προγραμματισμού C++.
- **Q#** (Q Sharp), κυκλοφόρησε από την Microsoft ως μέρος του **Quantum Development Kit**. Χρησιμοποιείται για τη σύνταξη υπορουτινών που εκτελούνται σε έναν κβαντικό επεξεργαστή, υπό τον έλεγχο ενός κλασσικού προγράμματος. Προς το παρόν, οι υπορουτίνες Q# εκτελούνται σε έναν προσομοιωτή ο οποίος παρέχεται από το Quantum Development Kit.

Ορισμένα παραδείγματα SDKs:

- **Qiskit,** υλοποιήθηκε από την IBM. Χρησιμοποιεί κυρίως την γλώσσα προγραμματισμού Python. Παρέχει εργαλεία για τη δημιουργία κβαντικών προγραμμάτων. Η εκτέλεση των κυκλωμάτων πραγματοποιείται σε προσομοιωτές αλλά και σε πραγματικούς κβαντικούς υπολογιστές.
- **Cirq,** υλοποιήθηκε από την Google. Χρησιμοποιεί τη γλώσσα προγραμματισμού Python για τη δημιουργία κβαντικών κυκλωμάτων. Τα αποτελέσματα αποκτώνται χρησιμοποιώντας προσομοιωτές που λειτουργούν στη συσκευή του χρήστη.
- **Quantum Development Kit,** υλοποιήθηκε από την Microsoft. Χρησιμοποιεί την γλώσσα προγραμματισμού Q#. Παρέχει ένα πλήρες περιβάλλον για την ανάπτυξη κβαντικών κυκλωμάτων. Η εκτέλεση των κυκλωμάτων πραγματοποιείται μόνο σε προσομοιωτές.

Στα επόμενα κεφάλαια θα αναλυθούν και θα συγκριθούν τα SDKs **Quantum Development Kit** της Microsoft και **Qiskit** της IBM.

4 Qiskit

4.1 Εισαγωγικά

Το **Qiskit (Quantum Information Science Kit)** κυκλοφόρησε στις 19 Δεκεμβρίου του 2018 και είναι ένα κιτ ανάπτυξης κβαντικού λογισμικού ανοικτού κώδικα (Open Source). Το Qiskit δημιουργήθηκε από την **IBM Research** δίνοντας έτσι την δυνατότητα ανάπτυξης λογισμικού για τις quantum cloud computing υπηρεσίες της. Επιτρέπει στους χρήστες να σχεδιάζουν κβαντικά κυκλώματα και να τα υλοποιούν σε προσομοιωτές αλλά ακόμα και σε αληθινούς κβαντικούς υπολογιστές οι οποίοι βρίσκονται διαθέσιμοι στο cloud και οι οποίοι παρέχονται από την **IBM Q**. Η IBM Q είναι μία βιομηχανία η οποία δημιουργήθηκε με σκοπό την κατασκευή κβαντικών υπολογιστών για επιχειρήσεις αλλά και για επιστημονική έρευνα. Μέσω της online πλατφόρμας **IBM Q Experience** παρέχεται η δυνατότητα στους χρήστες της να έχουν πρόσβαση στους κβαντικούς επεξεργαστές της IBM μέσω του cloud, ένα διαδικτυακό φόρουμ για συζήτηση σχετικών θεμάτων κβαντικής υπολογιστικής, tutorials για τον προγραμματισμό των IBM Q συσκευών καθώς και επιπρόσθετο εκπαιδευτικό υλικό σχετικά με τον κβαντική υπολογιστική.

Εικόνα 4-1. Λογότυπο Qiskit

Η παρούσα έκδοση του Qiskit χρησιμοποιεί την γλώσσα προγραμματισμού Python κατά κύριο λόγο, ενώ υπάρχουν και διαθέσιμες εκδόσεις που χρησιμοποιούν τις γλώσσες προγραμματισμού Swift και Java. Τα κβαντικά κυκλώματα που υλοποιούνται βασίζονται στην αναπαράσταση κβαντικών κυκλωμάτων της **OpenQASM** (**Open Quantum Assembly Language)**. Η OpenQASM είναι μία *συμβολική γλώσσα* προγραμματισμού η οποία περιγράφει τα κβαντικά κυκλώματα. Αποτελεί μία *ενδιάμεση αναπαράσταση (Intermediate Representation)* των κβαντικών εντολών [10].

Το Qiskit υποστηρίζει εκδόσεις της Python από την 3.5 και έπειτα. Προτείνεται στους χρήστες η εγκατάσταση του **Anaconda**, μια cross-platform διανομή της Python που χρησιμοποιείται κυρίως για επιστημονικές εφαρμογές.

Το **Jupyter Notebook**, που περιλαμβάνεται στο Anaconda, συνιστάται για αλληλεπίδραση με το Qiskit. Το Jupyter Notebook είναι μία εφαρμογή η οποία επιτρέπει την επεξεργασία και την εκτέλεση αρχείων **Notebook** μέσω ενός web browser. Τα αρχεία **Notebook** είναι αρχεία που δημιουργούνται από την εφαρμογή Jupyter Notebook, τα οποία περιλαμβάνουν κώδικα (π.χ. Python, Julia, R), οπτικοποιήσεις δεδομένων καθώς και στοιχεία κειμένου, όπως παράγραφοι, εξισώσεις, links κτλ.

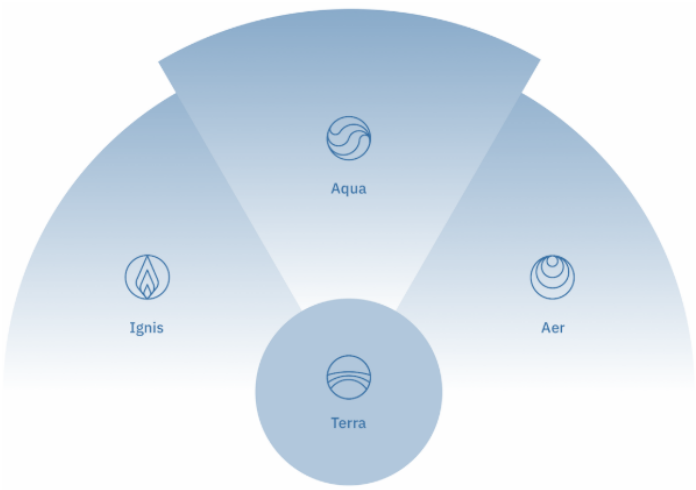
Το Qiskit υποστηρίζεται στα εξής 64-bit συστήματα:

- Windows 7 και έπειτα
- Ubuntu 16.04 και έπειτα
- macOS 10.12.6 και έπειτα

4.2 Qiskit Elements

Το Qiskit αποτελείται από τέσσερα θεμελιώδη στοιχεία, βασισμένα στα τέσσερα στοιχεία της φύσης, Γη, Αέρας, Νερό και Φωτιά.

- Terra
- Aer
- Aqua
- Ignis



Εικόνα 4-2. Θεμελιώδη στοιχεία του Qiskit

4.2.1 Terra

Το στοιχείο Terra παρέχει τα βασικά δομικά στοιχεία που είναι απαραίτητα για τον προγραμματισμό κβαντικών υπολογιστών. Παρέχει δηλαδή το υπόβαθρο για την σύνταξη των κβαντικών προγραμμάτων σε επίπεδο κυκλωμάτων. Τα κβαντικά κυκλώματα τρέχουν σε κάποιο από τα διαθέσιμα **backend**. Τα backends αντιπροσωπεύουν έναν προσομοιωτή ή έναν πραγματικό κβαντικό υπολογιστή και είναι υπεύθυνα για την εκτέλεση των κβαντικών κυκλωμάτων και την επιστροφή των αποτελεσμάτων. Πιο συγκεκριμένα, για την εκτέλεση ενός κβαντικού κυκλώματος απαιτούνται τέσσερα βασικά στοιχεία:

1. **Provider:** είναι μία οντότητα η οποία παρέχει την δυνατότητα πρόσβασης σε διαφορετικά backends. Γνωρίζει ανά πάσα στιγμή ποια backends είναι διαθέσιμα και αλληλεπιδρά μεταξύ τους. Το Qiskit επικοινωνεί με δύο Providers:
    a. **Aer**: παρέχει πρόσβαση σε προσομοιωτές που περιλαμβάνονται στο Qiskit και λειτουργούν στο τοπικό υπολογιστή.
    b. **IBM Q**: παρέχει πρόσβαση σε backends που βρίσκονται στο cloud της IBM Q. Μπορεί να είναι προσομοιωτές ή πραγματικές κβαντικές συσκευές.
2. **Backend:** όπως έχει ήδη αναφερθεί, τα backends μπορεί να είναι είτε ένας προσομοιωτής είτε ένας πραγματικός κβαντικός υπολογιστής, ο οποίος θα χρησιμοποιηθεί για την εκτέλεση μίας εργασίας.
3. **Job:** Το Job είναι ένα *instance* το οποίο μπορεί ανά πάση στιγμή να γνωρίζει την κατάσταση της υποβληθείσας εργασίας (για παράδειγμα, αν είναι *queued*, *running* ή *failed*) και επιτρέπει επίσης την δυνατότητα ελέγχου της εργασίας.
4. **Result:** Μόλις τελειώσει η εργασία, το στοιχείο Terra καθιστά δυνατή την ανάκτηση των αποτελεσμάτων από τα backends μέσω της μεθόδου «*job.result()*», η έξοδος της οποίας περιλαμβάνει τα κβαντικά δεδομένα. Ο πιο συνηθισμένος τρόπος αλληλεπίδρασης με τα δεδομένα είναι χρησιμοποιώντας την μέθοδο «*result.get_counts(quantum_circuit)*». Αυτή η μέθοδος επιστρέφει τις μετρήσεις για κάθε πιθανό αποτέλεσμα, οι οποίες έπειτα μπορούν να χρησιμοποιηθούν για περαιτέρω ανάλυση με εργαλεία που παρέχονται από το Terra.

Το στοιχείο Terra παρέχει επίσης εργαλεία για την απεικόνιση των κβαντικών κυκλωμάτων καθώς και εργαλεία για την απεικόνιση των αποτελεσμάτων σε διαγράμματα.

4.2.2 Aer

Το στοιχείο Aer βοηθάει στην κατανόηση των ορίων των κλασσικών επεξεργαστών, επιδεικνύοντας σε ποιο βαθμό μπορούν να μιμηθούν τους κβαντικούς υπολογισμούς. Το στοιχείο Aer παρέχει προσομοιωτές μέσω των οποίων μπορούν να τρέξουν τα κβαντικά κυκλώματα τα οποία έχουν υλοποιηθεί προηγουμένως στο στοιχείο Terra. Περιλαμβάνει τρεις προσομοιωτές υψηλής απόδοσης.

- **Qasm Simulator**: Επιτρέπει την ιδανική πολλαπλή εκτέλεση των κβαντικών κυκλωμάτων και επιστρέφει τις μετρήσεις για κάθε πιθανή κατάσταση.
- **Statevector Simulator:** Επιτρέπει την ιδανική single-shot εκτέλεση των κβαντικών κυκλωμάτων και επιστρέφει την τελική κβαντική κατάσταση του συστήματος ως ένα μιγαδικό διάνυσμα (**Statevector**) 2^n διαστάσεων, όπου *n* ο αριθμός των qubits.
- **Unitary Simulator:** Επιτρέπει την ιδανική single-shot εκτέλεση των κβαντικών κυκλωμάτων και επιστρέφει τον τελικό $2^n \times 2^n$ μοναδιακό πίνακα που αντιπροσωπεύει τις κβαντικές πύλες του κβαντικού κυκλώματος, όπου *n* ο αριθμός των qubits.

4.2.3 Ignis

Το στοιχείο Ignis χρησιμοποιείται για την καταπολέμηση του θορύβου και των σφαλμάτων. Περιλαμβάνει καλύτερο χαρακτηρισμό των σφαλμάτων, βελτίωση των κβαντικών πυλών και υλοποίηση κβαντικών υπολογισμών εν παρουσία θορύβου. Το στοιχείο Ignis απευθύνεται περισσότερο για όσους θέλουν να σχεδιάσουν κώδικες για διόρθωση σφαλμάτων ή επιθυμούν να μελετήσουν τρόπους χαρακτηρισμού των σφαλμάτων ή αναζητούν αποτελεσματικότερους τρόπους για τη χρήση των κβαντικών πυλών.

4.2.4 Aqua

Το στοιχείο Aqua χρησιμοποιείται για την υλοποίηση αλγορίθμων για εφαρμογές στον πραγματικό κόσμο. Το Aqua απευθύνεται σε ειδικούς στους τομείς της Χημείας, της Βελτιστοποίησης, της Οικονομίας και της Τεχνητής Νοημοσύνης, οι οποίοι επιθυμούν να διερευνήσουν τα οφέλη της χρήσης κβαντικών υπολογιστών προκειμένου να επιταχύνουν συγκεκριμένες υπολογιστικές εργασίες.

4.3 Αναπαράσταση Κβαντικών Καταστάσεων

Είναι σημαντικό να αναφερθεί ότι το Qiskit χρησιμοποιεί έναν διαφορετικό τρόπο αναπαράστασης της κβαντικής κατάστασης ενός συστήματος πολλαπλών qubit, από ότι έχει ήδη αναφερθεί και συνηθίζεται στα βιβλία φυσικής. Έστω *n* qubits, και έστω ότι το i-οστο qubit συμβολίζεται ως Q_i. Το Qiskit αναπαριστά την κβαντική κατάσταση των *n* qubits, με το τελευταίο qubit να βρίσκεται στο αριστερό μέρος και το πρώτο qubit να βρίσκεται στο δεξί μέρος κατά τον εξής τρόπο:

$$Q_{n-1} \otimes \ldots \otimes Q_1 \otimes Q_0$$

Για παράδειγμα, η κβαντική κατάσταση $|0\rangle \otimes |0\rangle \otimes |1\rangle = |001\rangle$ θα αναπαρασταθεί από το Qiskit ως $|1\rangle \otimes |0\rangle \otimes |0\rangle = |100\rangle$.

Αυτός ο τρόπος αναπαράστασης επηρεάζει και τον τρόπο αναπαράστασης των κβαντικών πυλών για πολλαπλά qubit. Για παράδειγμα μία πύλη CNOT για δύο qubit, με το πρώτο qubit ως control qubit και το δεύτερο qubit ως target qubit, αναπαρίσταται στο Qiskit ως εξής:

$$CNOT = (1\ 0\ \ 0\ 0\ \ 0\ 0\ \ 0\ 0\ \ 0\ 1\ \ \ \ 0\ 1\ 1\ 0\ \ 0\ 0\ \ )$$

4.4 Επισκόπηση Qiskit

Η βασική ροή εργασιών που χρησιμοποιεί το Qiskit χωρίζεται σε δύο στάδια.

- **Build**: το στάδιο της δημιουργίας του κβαντικού κυκλώματος
- **Execute**: το στάδιο εκτέλεσης των κυκλωμάτων σε διαφορετικά backends

Αφού πραγματοποιηθεί η εκτέλεση, τα αποτελέσματα συλλέγονται και μπορούν να υποβληθούν σε επεξεργασία.

Ακολουθεί ένα παράδειγμα δημιουργίας και μέτρησης της κβαντικής κατάστασης Bell $|\Phi^+\rangle$, προκειμένου να παρουσιαστούν τα βασικά στοιχεία προγραμματισμού του Qiskit.

4.4.1 Δημιουργία Κβαντικού Κυκλώματος

Το κύκλωμα χωρίζεται σε δύο **καταχωρητές (registers)**, το **quantum register** και το **classical register.** Στο quantum register αντιστοιχούν τα qubit και εκεί καταχωρούνται όλες οι κβαντικές πύλες. Στο classical register αντιστοιχούν τα bit και εκεί αποθηκεύονται τα αποτελέσματα των μετρήσεων.

Τα βασικά στοιχεία που χρειάζεται να εισαχθούν κατά την συγγραφή ενός προγράμματος είναι τα *QuantumRegister*, *ClassicalRegister*, *QuantumCircuit*.

```
In [1]: from qiskit import  QuantumRegister, ClassicalRegister, QuantumCircuit
```

Εικόνα 4-3. Qiskit Imports

Κατά την δημιουργία των registers χρειάζονται οι εξής εντολές:

```
In [2]: q = QuantumRegister(2)
        c = ClassicalRegister(2)
        qc = QuantumCircuit(q, c)
```

Εικόνα 4-4. Αρχικοποίηση κβαντικού κυκλώματος

- Η εντολή «***qr=QuantumRegister(2)***» δημιουργεί δύο qubit αρχικοποιημένα στην κατάσταση $|0\rangle$.
- Η εντολή «***cr=ClassicalRegister(2)***» δημιουργεί δύο bit στα οποία θα αποθηκευτούν τα αποτελέσματα των μετρήσεων για κάθε qubit αντίστοιχα.
- Η εντολή «***qc = QuantumCircuit(qr, cr)***» δημιουργεί το κβαντικό κύκλωμα με περιεχόμενα το quantum register (με δύο qubit) και το classical register (με δύο bit).

Οι παραπάνω εντολές, για λόγους συντομίας θα μπορούσαν να γραφτούν ως μία εντολή με τον εξής τρόπο: «***qc = QuantumCircuit(2, 2)***», όπου οι δύο αριθμοί στην παρένθεση υποδηλώνουν το μέγεθος του quantum register και το μέγεθος του classical register αντίστοιχα.

4.4.2 Υλοποίηση Κβαντικών Πυλών

Όπως έχει ήδη αναφερθεί, για την δημιουργία της κατάστασης Bell, $|\Phi^+\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, χρειάζονται δύο βήματα.

- Εφαρμογή της κβαντικής πύλης Hadamard στο πρώτο qubit
- Εφαρμογή της κβαντικής πύλης CNOT στα δύο qubits, με το πρώτο qubit ως control qubit και το δεύτερο qubit ως target qubit.

Οι εξής εντολές δίνονται ως εξής:

```
In [3]: qc.h(0)
        qc.cx(0,1)
```

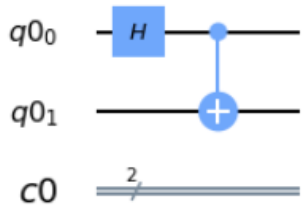
Εικόνα 4-5. Εφαρμογή κβαντικών πυλών

- Η εντολή «***qc.h(0)***» εφαρμόζει στο πρώτο qubit (όπως υποδηλώνει ο αριθμός 0 στην παρένθεση) του κβαντικού κυκλώματος qc την πύλη Hadamard.
- Η εντολή «***qc.cx(0,1)***», εφαρμόζει στο κβαντικό κύκλωμα qc την πύλη CNOT με control qubit το πρώτο qubit και target qubit το δεύτερο qubit (όπως υποδηλώνουν οι αριθμοί 0 και 1 αντίστοιχα στην παρένθεση). Ο πρώτος αριθμός στην παρένθεση υποδηλώνει ποιο qubit θα χρησιμοποιηθεί ως control qubit και ο δεύτερος αριθμός στην παρένθεση υποδηλώνει ποιο qubit θα χρησιμοποιηθεί ως target qubit.

4.4.3 Αναπαράσταση Κβαντικού Κυκλώματος

Το Qiskit παρέχει την δυνατότητα αναπαράστασης του κβαντικού κυκλώματος με την χρήση της εξής εντολής:

```
In [4]: qc.draw()
```



Εικόνα 4-6. Αναπαράσταση κβαντικού κυκλώματος

Τα ***q1₀*** και ***q1₁*** αντιστοιχούν στο πρώτο και στο δεύτερο qubit αντίστοιχα και αποτελούν το quantum register του κβαντικού κυκλώματος. Στο quantum register παρατηρούνται οι κβαντικές πύλες που εφαρμόστηκαν (Hadamard και CNOT) και εμφανίζονται με την ίδια σειρά με την οποία προγραμματίστηκαν (πρώτα η πύλη Hadamard και έπειτα η πύλη CNOT).

Το ***c1*** αποτελεί το classical register και ο αριθμός «2» στα δεξιά του υποδηλώνει πόσα bit έχουν δημιουργηθεί (δύο bit στην συγκεκριμένη περίπτωση).

4.4.4 Μέτρηση Κβαντικού Κυκλώματος

Η εντολή για την εφαρμογή της κβαντικής πύλης μέτρησης είναι της εξής μορφής:

```
In [5]: qc.measure([0,1],[0,1])
```

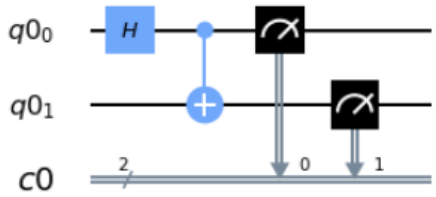
Εικόνα 4-7. Μέτρηση των qubits

Η εντολή «***qc.measure([0,1], [0,1])***» εφαρμόζει και στα δύο qubit από μία κβαντική πύλη μέτρησης. Συγκεκριμένα, το πρώτο μέρος της παρένθεσης «**[0,1]**, [0,1]» υποδηλώνει σε ποια qubit θα εφαρμοστεί η πύλη μέτρησης, όπου στην συγκεκριμένη περίπτωση θα εφαρμοστεί και στα δύο qubit. Το δεύτερο μέρος της παρένθεσης «[0,1], **[0,1]**» υποδηλώνει σε ποια bit θα αποθηκευτούν τα αποτελέσματα για το κάθε qubit αντίστοιχα. Επομένως, το αποτέλεσμα της μέτρησης του πρώτου qubit θα αποθηκευτεί στο πρώτο bit («[**0**,1],[**0**,1]») και το αποτέλεσμα της μέτρησης του δεύτερου qubit θα αποθηκευτεί στο δεύτερο bit («[0,**1**],[0,**1**]»).

Εναλλακτικά η παραπάνω εντολή θα μπορούσε να γραφτεί ως «***qc.measure(range(2),range(2))***».

Εφαρμόζοντας άλλη μια φορά την μέθοδο ***draw()***, το τελικό κβαντικό κύκλωμα είναι της εξής μορφής:

```
In [6]: qc.draw()
```



Εικόνα 4-8. Αναπαράσταση κβαντικού κυκλώματος με μετρήσεις

Μετά την υλοποίηση μίας μέτρησης στο Qiskit, τα αποτελέσματα καταχωρούνται στο classical register. Επομένως, η ανάκτηση των δεδομένων πραγματοποιείται με ανάγνωση των bit στο classical register. Όπως φαίνεται πλέον και στο παραπάνω σχήμα, το αποτέλεσμα της μέτρησης του πρώτου qubit θα καταχωρηθεί στο πρώτο bit του classical register (όπως υποδεικνύει ο αριθμός «0») και το αποτέλεσμα της μέτρησης του δεύτερου qubit θα καταχωρηθεί στο δεύτερο bit του classical register (όπως υποδεικνύει ο αριθμός «1»).

Σε αυτό το σημείο έχει ολοκληρωθεί το στάδιο Build του Qiskit. Ακολουθεί το στάδιο Execute κατά το οποίο θα πραγματοποιηθεί η εκτέλεση του κβαντικού κυκλώματος σε ένα backend.

4.4.5 Εκτέλεση Κβαντικού Κυκλώματος σε Προσομοιωτή

Για την εκτέλεση ενός κυκλώματος είναι απαραίτητη η χρήση ενός backend. Στο συγκεκριμένο παράδειγμα θα χρησιμοποιηθεί ο Aer provider που περιλαμβάνεται στο Qiskit και το backend που θα χρησιμοποιηθεί θα είναι ο "*qasm simulator*". Επιπλέον, θα χρειαστεί να εισαχθεί η μέθοδος execute η οποία εκτελεί το κύκλωμα σε ένα backend.

Ο κώδικας για την εκτέλεση του κβαντικού κυκλώματος δίνεται ως εξής:

```
In [7]: from qiskit import execute, Aer
        backend = Aer.get_backend('qasm_simulator')
        job = execute(qc, backend, shots=1000)
        result = job.result()
```

Εικόνα 4-9. Εκτέλεση κβαντικού κυκλώματος

- Η εντολή «***from qiskit import execute, Aer***" εισάγει τη μέθοδο *execute* και το στοιχείο *Aer*.
- Η εντολή «***backend = Aer.get_backend('qasm_simulator')***» επιλέγει τον "*qasm simulator*", από το στοιχείο Aer, ως backend και αποθηκεύεται στην μεταβλητή «backend».
- Η εντολή «***job = execute(qc, backend, shots=1000)***» εκτελεί το κβαντικό κύκλωμα qc, στο backend "*qasm simulator*", τρέχοντας το 1000 φορές προκειμένου να συλλεχθούν οι μετρήσεις

για κάθε πιθανή κατάσταση (η default τιμή για το πεδίο *shots* είναι 1024). Τα αποτελέσματα αποθηκεύονται στην μεταβλητή «*job*».

- Η εντολή «***result = job.result()***» ανακτά τα αποτελέσματα της εκτέλεσης από το *job* και τα αποθηκεύει στην μεταβλητή «result».

Προκειμένου να συλλεχθούν οι μετρήσεις χρησιμοποιείται η μέθοδος «***result.get_counts()***» και έπειτα η εντολή «***print()***» για την εμφάνισή τους.

```
In [8]:
counts = result.get_counts(qc)
print(counts)
```

```
{'11': 491, '00': 509}
```

Εικόνα 4-10. Εκτύπωση αποτελεσμάτων

Όπως είναι γνωστό, μία μέτρηση στην κατάσταση Bell $|\Phi^+\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, θα έχει ως αποτέλεσμα είτε τα δύο qubit να βρίσκονται στην κατάσταση $|00\rangle$, είτε τα δύο qubit να βρίσκονται στην κατάσταση $|11\rangle$, με ίση πιθανότητα. Εκτελώντας αυτή τη διαδικασία 1000 φορές, εκτιμάται ότι περίπου τις μισές φορές η κατάσταση που θα μετρηθεί θα είναι η κατάσταση $|00\rangle$ και τις υπόλοιπες μισές η κατάσταση που θα μετρηθεί θα είναι η κατάσταση $|11\rangle$.

Οι μετρήσεις που προέκυψαν από το συγκεκριμένο παράδειγμα, δείχνουν ότι από τις 1000 φορές που εκτελέστηκε το κβαντικό κύκλωμα:

- 509 φορές βρέθηκε στην κατάσταση $|00\rangle$ (Πιθανότητα 50.9%).
- 491 φορές βρέθηκε στην κατάσταση $|11\rangle$ (Πιθανότητα 49.1%).

Τα αποτελέσματα τα οποία παρατηρήθηκαν επιβεβαιώνουν την εκτίμηση των μετρήσεων, ότι δηλαδή περίπου τις μισές φορές θα μετρηθεί η κατάσταση $|00\rangle$ και τις υπόλοιπες η κατάσταση $|11\rangle$.

Δοκιμάζοντας, την εκτέλεση του ίδιου κυκλώματος, αυτή την φορά ένα εκατομμύριο φορές αντί για χίλιες φορές, αναμένεται ότι τα αποτελέσματα θα είναι πιο κοντά στο 50% για τις δύο καταστάσεις.

```
In [9]:
second_job = execute(qc, backend, shots=1000000)
second_result = second_job.result()
second_counts  = second_result.get_counts(qc)
print(second_counts)
```

```
{'11': 498547, '00': 501453}
```

Εικόνα 4-11. Εκτέλεση κβαντικού κυκλώματος

Οι μετρήσεις που προέκυψαν από την δεύτερη εκτέλεση είναι οι εξής:

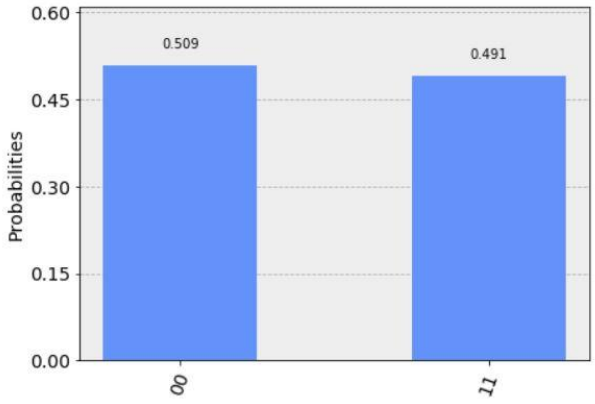
- 50.1% πιθανότητα στην κατάσταση $|00\rangle$.

- 49.9% πιθανότητα στην κατάσταση $|11\rangle$.

4.4.6 Απεικόνιση Αποτελεσμάτων σε Ιστόγραμμα

Το Qiskit παρέχει εργαλεία για την απεικόνιση των αποτελεσμάτων όπως για παράδειγμα εργαλεία για τη δημιουργία ιστογράμματος πιθανοτήτων των κβαντικών καταστάσεων. Εισάγοντας τα αποτελέσματα των μετρήσεων στην μέθοδο «***plot_histogram()***», προκύπτει το ιστόγραμμα πιθανοτήτων εμφάνισης των κβαντικών καταστάσεων.

```
In [13]: from qiskit.visualization import plot_histogram
         plot_histogram(counts)
```

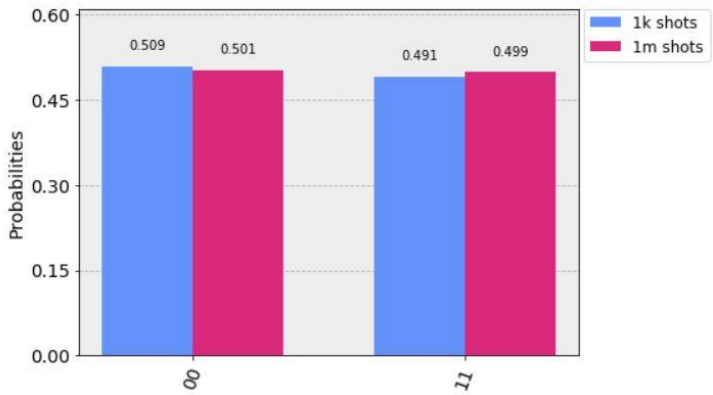


Εικόνα 4-12. Ιστόγραμμα κβαντικών καταστάσεων

Το παραπάνω ιστόγραμμα παρουσιάζει τις πιθανότητες εμφάνισης των κβαντικών καταστάσεων κατά την πρώτη εκτέλεση του κβαντικού κυκλώματος με τις χίλιες δοκιμές.

Χρησιμοποιώντας, την ίδια μέθοδο, είναι εφικτή η σύγκριση μεταξύ των δύο διαφορετικών εκτελέσεων του κβαντικού κυκλώματος.

```
In [11]: legend = ['1k shots', '1m shots']
         plot_histogram([counts, second_counts], legend=legend)
```



Εικόνα 4-13. Ιστόγραμμα κβαντικών καταστάσεων διαφορετικών μετρήσεων

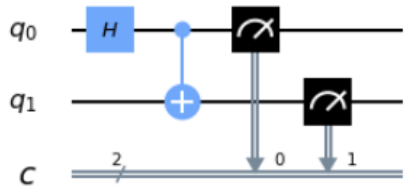
4.4.7 Εκτέλεση Κβαντικού Κυκλώματος σε Κβαντικό Υπολογιστή

Ένας χρήστης έχοντας λογαριασμό IBM Q, έχει την δυνατότητα να εκτελέσει το κύκλωμά του σε πραγματικές κβαντικές συσκευές ή και σε προσομοιωτές που παρέχονται μέσω του cloud. Για την πρόσβαση στις συσκευές IBM Q, χρειάζεται ένα API token το οποίο ένας συνδεδεμένος χρήστης μπορεί να παράξει μέσω της πλατφόρμας IBM Q Experience. Εφόσον έχει παράξει το API token του, πρέπει να τρέξει την εντολή «***IBMQ.save_account('API_TOKEN')***» (όπου στην θέση του 'API_TOKEN' εννοείται το token που αντιστοιχεί στον χρήστη) προκειμένου να αποθηκεύσει τα πιστοποιητικά IBM Q σε ένα τοπικό αρχείο. Εάν δεν έχουν αλλάξει τα στοιχεία εγγραφής του χρήστη, δεν χρειάζεται να εκτελέσει ξανά την συγκεκριμένη εντολή.

Ακολουθεί το ίδιο παράδειγμα δημιουργίας και μέτρησης της κβαντικής κατάστασης Bell $|\Phi^+\rangle$, με την διαφορά ότι αυτή τη φορά η εκτέλεση του κυκλώματος θα πραγματοποιηθεί σε έναν από τους πραγματικούς κβαντικούς υπολογιστές στους οποίους παρέχει πρόσβαση η IBMQ. Ο παρακάτω κώδικας αφορά το στάδιο Build και είναι κατά τον μεγαλύτερο βαθμό ο ίδιος με τον κώδικα που χρησιμοποιήθηκε προηγουμένως για την δημιουργία του κβαντικού κυκλώματος, με μικρές αλλαγές για λόγους συντομίας.

```
In [1]:
from qiskit import QuantumCircuit
from qiskit.visualization import *

qc = QuantumCircuit(2, 2)
qc.h(0)
qc.cx(0,1)
qc.measure(range(2),range(2))
qc.draw()
```



Εικόνα 4-14. Δημιουργία και μέτρηση κβαντικής κατάστασης Bell

Έχοντας τελειώσει το στάδιο Build, κατά το στάδιο Execute ο χρήστης πρέπει να φορτώσει τον λογαριασμό του προκειμένου να έχει πρόσβαση στις υπηρεσίες της IBM Q.

```
In [2]:
from qiskit import IBMQ

IBMQ.load_account()
IBMQ.providers()
```

```
[<AccountProvider for IBMQ(hub='ibm-q', group='open', project='main')>]
```

Εικόνα 4-15. Φόρτωση λογαριασμού IBM Q και λίστα με providers

- Η εντολή «***from qiskit import IBMQ***" παρέχει πρόσβαση στις υπηρεσίες IBMQ.
- Η εντολή «***IBMQ.load_account()***» φορτώνει τον λογαριασμό του χρήστη, έχοντας βέβαια ήδη αποθηκεύσει πρωτύτερα τον λογαριασμό του.
- Η εντολή «***IBMQ.providers()***» επιστρέφει μία λίστα με τους διαθέσιμους providers της IBM Q τους οποίους μπορεί να χρησιμοποιήσει ο χρήστης προκειμένου να έχει πρόσβαση στα διαφορετικά backends. Στο συγκεκριμένο παράδειγμα, η λίστα περιλάμβανε μόνο έναν διαθέσιμο provider.

Έπειτα ο χρήστης επιλέγει τον provider που επιθυμεί μέσω των πεδίων *hub*, *group*, *project* ή και με ένα συνδυασμό τους. Στον παρακάτω κώδικα επιλέγεται ο provider μέσω του πεδίου *group* ο οποίος αποθηκεύεται στην μεταβλητή «provider».

```
In [3]: provider = IBMQ.get_provider(group='open')
```

Εικόνα 4-16. Επιλογή provider

Στη συνέχεια, ο χρήστης μέσω της μεθόδου «***backends()***» λαμβάνει μία λίστα με τα διαθέσιμα backends.

```
In [4]: provider.backends()
```

```
[<IBMQSimulator('ibmq_qasm_simulator') from IBMQ(hub='ibm-q', group='open', project='main')>,
 <IBMQBackend('ibmqx2') from IBMQ(hub='ibm-q', group='open', project='main')>,
 <IBMQBackend('ibmq_16_melbourne') from IBMQ(hub='ibm-q', group='open', project='main')>,
 <IBMQBackend('ibmq_vigo') from IBMQ(hub='ibm-q', group='open', project='main')>,
 <IBMQBackend('ibmq_ourense') from IBMQ(hub='ibm-q', group='open', project='main')>]
```

Εικόνα 4-17. Λίστα με backends

Στο συγκεκριμένο παράδειγμα η λίστα περιλαμβάνει τέσσερα διαθέσιμα backends. Το πρώτο backend με το όνομα "ibmq_qasm_simulator" αποτελεί έναν προσομοιωτή. Τα υπόλοιπα backends, "*ibmqx4*", "*ibmq*_16_melbourne", "*ibmq_vigo*", " "*ibmq*_ourense", είναι πραγματικοί κβαντικοί υπολογιστές στους οποίους ο χρήστης μπορεί να εκτελέσει το κύκλωμα του.

Το qiskit μέσω της μεθόδου «***lambda***» επιτρέπει το φιλτράρισμα των backend, δίνοντας την δυνατότητα στον χρήστη να επιλέξει το backend που ταιριάζει περισσότερο στις απαιτήσεις του. Ο χρήστης μπορεί να επιλέξει backend σύμφωνα με τον αριθμό των qubit που περιέχει, αν είναι προσομοιωτής ή όχι καθώς και αν είναι λειτουργικός και όχι υπό συντήρηση.

```
provider.backends(filters=lambda x: x.configuration().n_qubits>=10        # more than 10 qubits
                                    and not x.configuration().simulator   # not a simulator
                                    and x.status().operational==True)     # operational backend
```

Εικόνα 4-18. Φιλτράρισμα backends

Στο παραπάνω παράδειγμα, ο χρήστης ζητάει την λίστα με τα backends τα οποία έχουν δέκα ή και περισσότερα qubits, είναι πραγματικές συσκευές και είναι λειτουργικά.

Μία άλλη χρήσιμη ιδιότητα, είναι ο χρήστης να μπορεί να γνωρίζει ποιο backend έχει την μικρότερη ουρά αναμονής. Όταν ένας χρήστης θέλει να εκτελέσει το κύκλωμά του σε ένα backend της IBM Q, μπαίνει σε μία ουρά αναμονής, ανάλογα με το πόσοι άλλοι χρήστες θέλουν να εκτελέσουν το δικό τους κύκλωμα στην ίδια συσκευή και όταν μόνο εξυπηρετηθούν όλοι οι χρήστες που έστειλαν το αίτημα εκτέλεσης πριν από αυτόν, ο χρήστης μπορεί να εκτελέσει το κύκλωμα του. Επομένως, όταν ένας χρήστης θέλει τα αποτελέσματά του όσο πιο άμεσα γίνεται θα ήταν χρήσιμο να γνωρίζει ποιες πραγματικές συσκευές έχουν την μικρότερη ουρά αναμονής. Την μεγαλύτερη ουρά αναμονής συνήθως την έχουν οι πραγματικοί κβαντικοί υπολογιστές και όχι οι προσομοιωτές. Το qiskit παρέχει την μέθοδο «***least_busy()***» η οποία επιστρέφει στον χρήστη το backend με την μικρότερη ουρά αναμονής.

Η εντολή για την εύρεση του κβαντικού υπολογιστή με την μικρότερη ουρά αναμονής είναι η εξής:

```
In [5]:
from qiskit.providers.ibmq import least_busy
backend = provider.backends(filters=lambda x: not x.configuration().simulator)
least_busy(backend)
```

```
<IBMQBackend('ibmq_16_melbourne') from IBMQ(hub='ibm-q', group='open', project='main')>
```

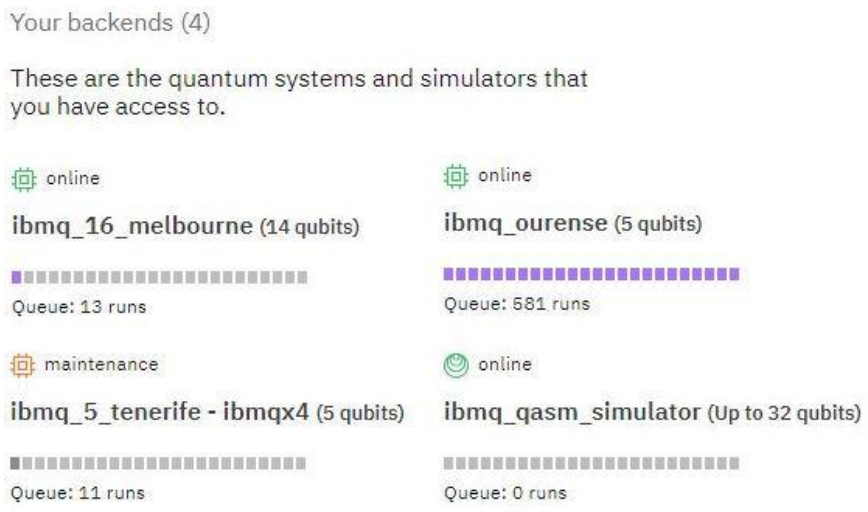
Εικόνα 4-19. Εύρεση κβαντικού υπολογιστή με την μικρότερη ουρά αναμονής

Επομένως, ο "*ibmq_16_melbourne*" είναι ο κβαντικός υπολογιστής με την μικρότερη ουρά αναμονής. Ο χρήστης μπορεί να επιλέξει το backend που επιθυμεί να χρησιμοποιήσει μέσω της εντολής «***get_backend('BACKEND_NAME')***» όπου στο πεδίο 'BACKEND_NAME' πληκτρολογεί το όνομα του διαθέσιμου backend.

```
In [6]:
backend = provider.get_backend('ibmq_16_melbourne')
```

Εικόνα 4-20. Επιλογή backend

Αξίζει να σημειωθεί ότι ο χρήστης μπορεί να δει ποια backends είναι διαθέσιμα και μέσω της πλατφόρμας IBM Q Experience μαζί με ορισμένες βασικές πληροφορίες για το κάθε backend, όπως το αν είναι λειτουργικά ή αν βρίσκονται υπό συντήρηση, τον αριθμό των qubit που περιέχουν και την ουρά αναμονής για κάθε backend.



Εικόνα 4-21. Διαθέσιμα backends

Για την εκτέλεση του κβαντικού κυκλώματος χρησιμοποιείται και πάλι η μέθοδος «***execute()***». Η εκτέλεση θα πραγματοποιηθεί 1024 φορές (default τιμή). Όπως έχει ήδη αναφερθεί, η εργασία που υποβάλλεται προστίθεται στο τέλος της ουράς αναμονής. Το qiskit παρέχει την μέθοδο «***job_monitor()***», προκειμένου ο χρήστης να παρακολουθεί (monitor) το status της εργασίας του. Κάθε φορά που αλλάζει η κατάσταση της εργασίας το *Job Status* ανανεώνεται.

```
In [7]:
from qiskit import execute
from qiskit.tools.monitor import job_monitor

job = execute(qc, backend=backend)
job_monitor(job)
```

```
Job Status: job has successfully run
```

Εικόνα 4-22. Εκτέλεση κβαντικού κυκλώματος

Στο συγκεκριμένο παράδειγμα, το *job_monitor* κατέγραψε τα εξής status:

1. **Job Status: job is being initialized** (αρχικοποίηση εργασίας)
2. **Job Status: job is queued (2)** (αριθμός ουράς αναμονής)
3. **Job Status: job is queued (1)** (αριθμός ουράς αναμονής)
4. **Job Status: job has successfully run** (ολοκλήρωση εργασίας)

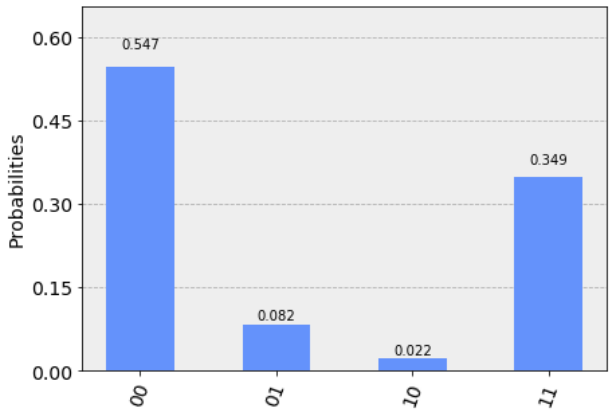
Έπειτα, όπως και προηγουμένως, με την εντολή «***result = job.result()***» ο χρήστης ανακτά τα αποτελέσματα της εκτέλεσης από το *job.*

```
In [8]:
result = job.result()
```

Εικόνα 4-23. Ανάκτηση αποτελεσμάτων

Σχεδιάζοντας το ιστόγραμμα, προκύπτουν τα εξής αποτελέσματα:

```
In [9]:
counts = result.get_counts(qc)
plot_histogram(counts)
```



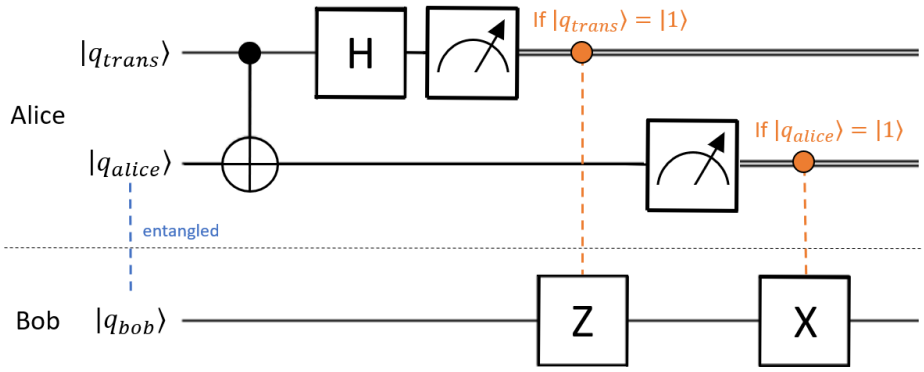
Εικόνα 4-24. Ιστόγραμμα κβαντικών καταστάσεων

Στο συγκεκριμένο ιστόγραμμα κβαντικών καταστάσεων παρατηρούνται και ορισμένα αποτελέσματα τα οποία δεν συμφωνούν με τα αναμενόμενα αποτελέσματα. Είναι γνωστό ότι μία μέτρηση στην κβαντική κατάσταση Bell $|\Phi^+\rangle$ θα έχει ως αποτέλεσμα μόνο μία από τις δύο καταστάσεις $|00\rangle$ ή $|11\rangle$ με ίση πιθανότητα. Πραγματοποιώντας 1024 αντίστοιχες μετρήσεις αναμένεται ότι περίπου τις μισές φορές το

αποτέλεσμα θα είναι η κατάσταση $|00\rangle$ και τις υπόλοιπες η κατάσταση $|11\rangle$. Στο παραπάνω ιστόγραμμα, παρατηρήθηκαν και ορισμένες περιπτώσεις όπου η κατάσταση του συστήματος μετά την μέτρηση ήταν $|01\rangle$ ή $|10\rangle$ (βέβαια με μικρότερη πιθανότητα από τις $|00\rangle$ και $|11\rangle$), γεγονός που αντικρούει την παραπάνω εκτίμηση. Το συγκεκριμένο γεγονός είναι αποτέλεσμα της Κβαντικής Αποσυνοχής. Από την στιγμή που το κβαντικό κύκλωμα εκτελείται σε πραγματικό κβαντικό υπολογιστή και όχι σε προσομοιωτή, είναι αναμενόμενη η ύπαρξη σφαλμάτων. Όπως έχει ήδη αναφερθεί, τα σφάλματα είναι εγγενή στα κβαντικά υπολογιστικά συστήματα. Το γεγονός όμως ότι οι καταστάσεις $|00\rangle$ και $|11\rangle$ είναι οι δύο πιο πιθανές, δείχνει ότι για ένα μικρό κύκλωμα όπως το συγκεκριμένο, ακόμα και με την ύπαρξη σφάλματος, είναι δυνατός ο σχηματισμός μίας εύστοχης εκτίμησης. Όσο όμως τα κυκλώματα γίνονται μεγαλύτερα και όσο ο αριθμός των qubit αυξάνεται, αυξάνεται και ο αριθμός των σφαλμάτων. Γίνεται πλέον φανερό και στην πράξη πως ο προγραμματισμός σε κβαντικά υπολογιστικά συστήματα, μέχρι και σήμερα, περιλαμβάνει σφάλματα τα οποία εμποδίζουν την πραγματοποίηση πολυπλοκότερων υπολογισμών.

4.5 Κβαντική Τηλεμεταφορά

Στην ενότητα αυτή, θα υλοποιηθεί ο αλγόριθμος για την κβαντική τηλεμεταφορά μέσω του Qiskit. Υπενθυμίζεται ότι για να επιτευχθεί η κβαντική τηλεμεταφορά ενός qubit q_{trans} από την Alice στον Bob, χρειάζεται να μοιράζονται μεταξύ τους ένα ζευγάρι διαπλεγμένων qubit (q_{alice} και q_{bob} αντίστοιχα). Παρακάτω παρατίθεται το κύκλωμα κβαντικής τηλεμεταφοράς για υπενθύμιση.



Εικόνα 4-25. Κύκλωμα Κβαντικής Τηλεμεταφοράς

Ξεκινώντας με το Qiskit, δημιουργείται ένα quantum register με τρία qubit τα οποία αντιπροσωπεύουν τα q_{trans}, q_{alice} και q_{bob}. Επιπλέον, δημιουργούνται τρία διαφορετικά classical registers μεγέθους ένα, στα οποία θα αποθηκεύονται τα αποτελέσματα των μετρήσεων των παραπάνω qubit.

Ο λόγος για τον οποίο αυτή τη φορά θα δημιουργηθούν τρία διαφορετικά classical registers μεγέθους ένα αντί για ένα classical register μεγέθους τρία, είναι γιατί στην πορεία θα χρειαστεί η μέθοδος «***c_if()***», προκειμένου να εφαρμοστούν ορισμένες πύλες υπό συνθήκη, η οποία δέχεται ως είσοδο ένα ολόκληρο classical register. Από τη στιγμή που χρειάζεται να ελεγχθούν μεμονωμένα bit, χρειάζεται να δημιουργηθεί ένα classical register για κάθε bit ξεχωριστά προκειμένου να εφαρμοστεί η συγκεκριμένη μέθοδος.

```
In [1]:
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister, Aer, execute

qr = QuantumRegister(3) # Create quantum register
# qr[0]:qtrans, qr[1]:qalice, qr[2]:qbob

c0 = ClassicalRegister(1) # Create three classical registers
c1 = ClassicalRegister(1)
c2 = ClassicalRegister(1)
qc = QuantumCircuit(qr,c0,c1,c2) # Create quantum circuit
```

Εικόνα 4-26. Δημιουργία quantum και classical register

Για λόγους απλοποίησης, αντί μιας τυχαίας κατάστασης $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, η κατάσταση η οποία επιλέγεται για τηλεμεταφορά στο συγκεκριμένο παράδειγμα θα είναι η κατάσταση $|\psi\rangle = |1\rangle = |q_{trans}\rangle$. Συνεπώς, μετά την εκτέλεση του κβαντικού κυκλώματος αναμένεται ότι $|q_{bob}\rangle = |1\rangle$.

Πριν την υλοποίηση του αλγορίθμου χρειάζονται τα δύο εξής αρχικά βήματα:

1. Αρχικοποίηση του qubit προς τηλεμεταφορά (στην συγκεκριμένη περίπτωση $|q_{trans}\rangle = |1\rangle$). Είναι γνωστό ότι κατά την δημιουργία του quantum register, όλα τα qubit αρχικοποιούνται στην κατάσταση $|0\rangle$. Επομένως, προκειμένου $|q_{trans}\rangle = |1\rangle$, θα χρειαστεί να εφαρμοστεί μία πύλη X στο q_{trans}.
2. Τα q_{alice} και q_{bob} πρέπει να είναι διαπλεγμένα. Επομένως, εφαρμογή πύλης Hadamard στο q_{alice} και έπειτα εφαρμογή πύλης CNOT με το q_{alice} ως control qubit και το q_{bob} ως target qubit.

```
In [2]:
# Set qtrans=|1>
qc.x(qr[0])

# Entangle qalice and qbob
qc.h(qr[1])
qc.cx(qr[1],qr[2])
```

Εικόνα 4-27. Αρχικοποίηση qtrans και κβαντική διεμπλοκή μεταξύ qalice και qbob

Σε αυτό το σημείο, έχοντας τελειώσει με τα δύο αρχικά βήματα, ξεκινάει το στάδιο υλοποίησης του αλγορίθμου της κβαντικής τηλεμεταφοράς. Τα βήματα του αλγορίθμου είναι τα εξής:

1. η Alice εφαρμόζει μία πύλη CNOT μεταξύ των q_{trans} και q_{alice} με το q_{trans} ως control qubit και το q_{alice} ως target qubit.
2. η Alice εφαρμόζει μία πύλη Hadamard στο q_{trans}.
3. η Alice πραγματοποιεί μία μέτρηση στο q_{trans}.
    a. Αν η μέτρηση δείξει την κατάσταση $|1\rangle$, τότε ο Bob εφαρμόζει μία πύλη Pauli Z στο q_{bob}.
4. η Alice πραγματοποιεί μία μέτρηση στο q_{alice}.
    a. Αν η μέτρηση δείξει την κατάσταση $|1\rangle$, τότε ο Bob εφαρμόζει μία πύλη Pauli X στο q_{bob}.

```
In [3]:
# Step 1 - Apply CNOT to qtrans and qalice
qc.cx(qr[0],qr[1])
# Step 2 - Apply Hadamard to qtrans
qc.h(qr[0])

# Step 3
qc.measure(qr[0],c0[0]) # Measure qtrans
qc.z(qr[2]).c_if(c0,1) # if qtrans=|1>, apply Z to qbob
# Step 4
qc.measure(qr[1],c1[0]) # Measure qalice
qc.x(qr[2]).c_if(c1,1) # if qalice=|1>, apply X to qbob

qc.measure(qr[2],c2[0]) # Measure qbob
```

Εικόνα 4-28. Αλγόριθμος κβαντικής τηλεμεταφοράς

Εκτελώντας το πρόγραμμα 1000 φορές αναμένεται ότι $|q_{bob}\rangle = |1\rangle$ σε όλες τις μετρήσεις που θα πραγματοποιηθούν. Χρησιμοποιώντας τον προσομοιωτή *qasm_simulator* προκύπτουν τα εξής αποτελέσματα:

```
In [4]:
backend = Aer.get_backend('qasm_simulator')
job = execute(qc, backend, shots=1000)
result = job.result()

counts = result.get_counts(qc)
print(counts)
```

```
{'1 0 1': 242, '1 1 1': 245, '1 0 0': 259, '1 1 0': 254}
```

Εικόνα 4-29. Εκτέλεση κβαντικού κυκλώματος και εκτύπωση αποτελεσμάτων

Όπως έχει ήδη αναφερθεί, η αναπαράσταση των κβαντικών καταστάσεων στο Qiskit είναι της μορφής $Q_{n-1} \otimes \ldots \otimes Q_1 \otimes Q_0$. Στο συγκεκριμένο παράδειγμα επομένως, η σειρά εμφάνισης των qubit θα είναι η εξής:

$$q_{bob} \otimes q_{alice} \otimes q_{trans}$$

Οι καταστάσεις που παρατηρήθηκαν στις εκτελέσεις του κυκλώματος είναι οι $|101\rangle$, $|111\rangle$, $|100\rangle$ και $|110\rangle$.

$$|101\rangle = |1\rangle \otimes |0\rangle \otimes |1\rangle = |q_{bob}\rangle \otimes |q_{alice}\rangle \otimes |q_{trans}\rangle \Rightarrow |q_{bob}\rangle = |1\rangle$$

$$|111\rangle = |1\rangle \otimes |1\rangle \otimes |1\rangle = |q_{bob}\rangle \otimes |q_{alice}\rangle \otimes |q_{trans}\rangle \Rightarrow |q_{bob}\rangle = |1\rangle$$

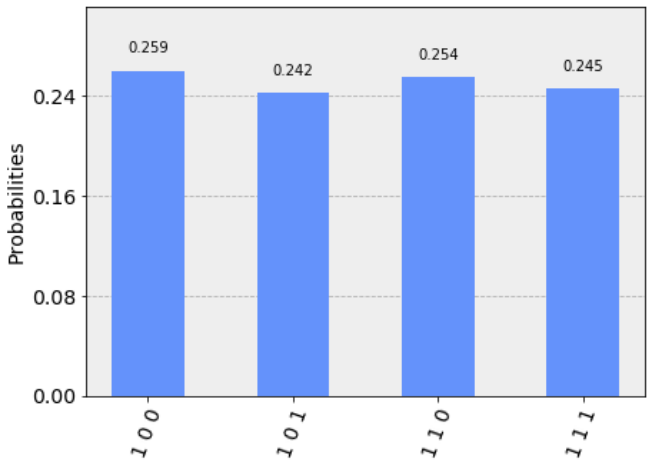
$$|100\rangle = |1\rangle \otimes |0\rangle \otimes |0\rangle = |q_{bob}\rangle \otimes |q_{alice}\rangle \otimes |q_{trans}\rangle \Rightarrow |q_{bob}\rangle = |1\rangle$$

$$|110\rangle = |1\rangle \otimes |1\rangle \otimes |0\rangle = |q_{bob}\rangle \otimes |q_{alice}\rangle \otimes |q_{trans}\rangle \Rightarrow |q_{bob}\rangle = |1\rangle$$

Πράγματι, σε κάθε περίπτωση $|q_{bob}\rangle = |1\rangle$, γεγονός που αποδεικνύει ότι η τηλεμεταφορά της κατάστασης $|1\rangle$ από το q_{trans} στο q_{bob} πραγματοποιήθηκε επιτυχώς.

Σχεδιάζοντας το ιστόγραμμα των πιθανών κβαντικών καταστάσεων:

```
In [5]: from qiskit.tools.visualization import plot_histogram
        plot_histogram(result.get_counts(qc))
```



Εικόνα 4-30. Ιστόγραμμα κβαντικών καταστάσεων

4.6 Αλγόριθμός Bernstein-Vazirani

Στην ενότητα αυτή, θα υλοποιηθεί ο αλγόριθμος Bernstein-Vazirani μέσω του Qiskit. Υπενθυμίζεται ότι δίνεται μία συνάρτηση $f : \{0,1\}^n \rightarrow \{0,1\}$ όπου:

$f(x) = u \cdot x \ (mod2) = (u_1 * x_1 + u_2 * x_2 + \ldots + u_n * x_n)(mod2)$, μίας άγνωστης συμβολοσειράς $u \in \{0,1\}^n$. Σκοπός του προβλήματος είναι η εύρεση της άγνωστης συμβολοσειράς u.

Στο συγκεκριμένο παράδειγμα, επιλέγεται ο ακέραιος αριθμός 101 ο οποίος αντιστοιχεί στον δυαδικό αριθμό 1100101, μεγέθους 7 bit, ο οποίος θα αντιπροσωπεύει την άγνωστη συμβολοσειρά u (δηλαδή $u = 1100101$).

```
In [1]:
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister, Aer, execute

nQubits = 7 # Number of qubits
u = 101     # Hidden integer whose bitstring is "1100101"

qr = QuantumRegister(nQubits)   # Create quantum register
cr = ClassicalRegister(nQubits) # Create classical register

qc = QuantumCircuit(qr, cr) # Create quantum circuit
```

Εικόνα 4-31. Ορισμός συμβολοσειράς u και δημιουργία quantum και classical register

Όπως φαίνεται στην παραπάνω εικόνα, ορίζεται ο αριθμός των qubit που θα χρησιμοποιηθούν (στο συγκεκριμένο παράδειγμα θα χρησιμοποιηθούν 7 qubit), ορίζεται η συμβολοσειρά u και δημιουργούνται τα quantum και classical registers μεγέθους εφτά τα οποία θα απαρτίζουν το κβαντικό κύκλωμα.

Υπενθυμίζεται ότι τα βήματα του αλγορίθμου, για την εύρεση της «άγνωστης» συμβολοσειράς u, είναι τα εξής:

1. Εφαρμογή πύλης Hadamard σε κάθε qubit.
2. Εφαρμογή μαύρου κουτιού (oracle) Q_f, το οποίο υλοποιεί την συνάρτηση f.
3. Εφαρμογή της πύλης Hadamard σε κάθε qubit.
4. Μέτρηση κάθε qubit.

```
In [2]:
# Step 1 - Apply Hadamard gates before querying the oracle
for i in range(nQubits):
    qc.h(qr[i])

# Apply barrier to prevent gate reordering by the compiler for optimization
qc.barrier()

# Step 2 - Apply the "black box"
for i in range(nQubits):
    if (u & (1 << (i))):
        qc.z(qr[6-i])
    else:
        qc.iden(qr[6-i])

# Apply barrier
qc.barrier()

# Step 3 - Apply Hadamard gates after querying the "black box"
for i in range(nQubits):
    qc.h(qr[i])

# Step 4 - Measurement
qc.barrier(qr)
qc.measure(qr, cr)
```

Εικόνα 4-32. Αλγόριθμος Bernstein-Vazirani

Η εντολή «***qc.barrier()***» εφαρμόζει ένα φράγμα (barrier) προκειμένου να αποτρέψει τον μεταγλωττιστή να πραγματοποιήσει αναδιάταξη των κβαντικών πυλών για λόγους βελτιστοποίησης. Επομένως, εφαρμόζοντας ένα φράγμα εξασφαλίζεται ότι η διάταξη των κβαντικών πυλών μέχρι εκείνο το σημείο θα παραμείνει με την σειρά την οποία έχουν προγραμματιστεί.

Κατά το δεύτερο βήμα του αλγορίθμου, εφαρμόζεται το «μαύρο κουτί» (ή αλλιώς ο μετασχηματισμός Q_f). Όπως έχει αναφερθεί, ο μετασχηματισμός αυτός αποσυντίθεται σε n-κβαντικές πύλες, όπου η κάθε μία δρα στο qubit που της αντιστοιχεί με τον εξής τρόπο:

$$Q_i = \{I \text{ , αν } u_i = 0 \; Z \text{ , αν } u_i = 1$$

Η συνθήκη «***if (u & (1 << i))***» ελέγχει αν το στοιχείο i της συμβολοσειράς u είναι 0 ή όχι. Οι τελεστές «**&**» και «**<<**» είναι *bitwise operators* της Python, οι οποίοι αντιμετωπίζουν μια τιμή ως μια σειρά από bit και όχι ως μια αριθμητική ποσότητα. Αντιμετωπίζουν τους αριθμούς δηλαδή στο δυαδικό σύστημα και όχι στο δεκαδικό, επομένως στο συγκεκριμένο παράδειγμα η τιμή 101 (δεκαδικό σύστημα) αντιμετωπίζεται ως 1100101 (δυαδικό σύστημα).

Πιο συγκεκριμένα, ο τελεστής «**<<**» πραγματοποιεί μία αριστερή μετατόπιση προσθέτοντας μηδενικά από δεξιά. Επομένως, το «***x<<y***» επιστρέφει το x με τα bits μετατοπισμένα προς τα αριστερά y θέσεις (και τα νέα bits στη δεξιά πλευρά συμπληρώνονται από μηδενικά). Στο συγκεκριμένο παράδειγμα, η εντολή «***1<< i***» με $0 \le i \le 6$, παράγει τους εξής αριθμούς:

$$i = 0 \rightarrow 1$$

$$i = 1 \rightarrow 10$$

$$i = 2 \rightarrow 100$$

$$i = 3 \rightarrow 1000$$

$$i = 4 \rightarrow 10000$$

$$i = 5 \rightarrow 100000$$

$$i = 6 \rightarrow 1000000$$

Σημειώνεται ότι τα bit strings που προέκυψαν, μπορούν να αναπαρασταθούν όλα ισοδύναμα ως bit strings των 7 bit (προκειμένου να έχουν τον ίδιο αριθμό bit με την συμβολοσειρά u) κατά τον εξής τρόπο:

$$i = 0 \rightarrow 1 = 0000001$$

$$i = 1 \rightarrow 10 = 0000010$$

$$i = 2 \rightarrow 100 = 0000100$$

$$\vdots$$

Ο τελεστής «**&**» πραγματοποιεί την λογική πράξη AND. Επομένως, η συνθήκη «***if (u & (1 << i))***» για $0 \le i \le 6$, παράγει τα εξής αποτελέσματα:

$$i = 0 \rightarrow 1100101 \;\&\; 0000001 \rightarrow 0000001 \text{ , έλεγχος } 7^{\text{ου}} \text{ bit}$$

$i = 1 \rightarrow 1100101 \ \& \ 0000010 \rightarrow 0000000$, έλεγχος 6ου bit

$i = 2 \rightarrow 1100101 \ \& \ 0000100 \rightarrow 0000100$, έλεγχος 5ου bit

$i = 3 \rightarrow 1100101 \ \& \ 0001000 \rightarrow 0000000$, έλεγχος 4ου bit

$i = 4 \rightarrow 1100101 \ \& \ 0010000 \rightarrow 0000000$, έλεγχος 3ου bit

$i = 5 \rightarrow 1100101 \ \& \ 0100000 \rightarrow 0100000$, έλεγχος 2ου bit

$i = 6 \rightarrow 1100101 \ \& \ 1000000 \rightarrow 1000000$, έλεγχος 1ου bit

Με αυτό τον τρόπο λοιπόν μπορεί να ελεγχθεί κάθε bit της συμβολοσειράς u μεμονωμένα προκειμένου να προσδιοριστεί αν είναι 0 ή 1.

Σημειώνεται ότι, για $i = 0$ πραγματοποιείται έλεγχος για το τελευταίο bit της συμβολοσειράς (και όχι για το πρώτο), επομένως η αντίστοιχη πύλη θα πρέπει να εφαρμοστεί στο τελευταίο qubit (και όχι στο πρώτο qubit). Για τον λόγο αυτό, για κάθε i η αντίστοιχη πύλη Q_i θα εφαρμόζεται στο qubit στη θέση 6-i.

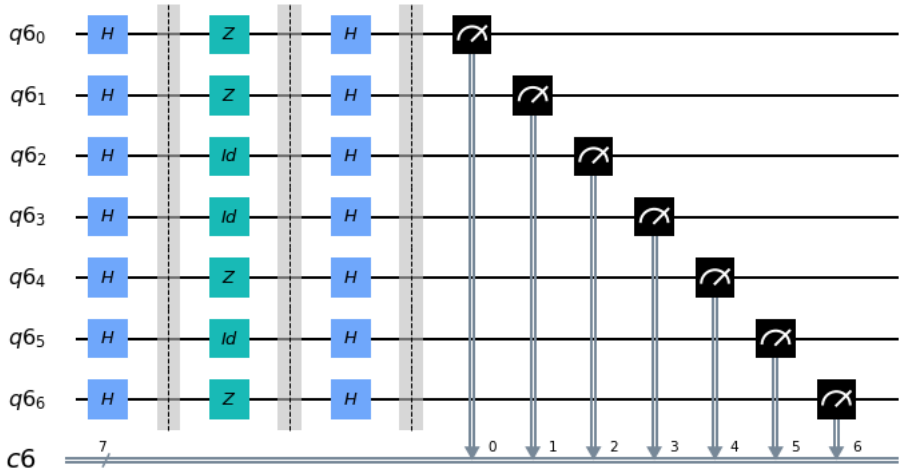
$$i = 0 \rightarrow Q_0|q_6\rangle \ i = 4 \rightarrow Q_4|q_2\rangle \ i = 1 \rightarrow Q_1|q_5\rangle \ i = 5 \rightarrow Q_5|q_1\rangle \ i = 2 \rightarrow Q_2|q_4\rangle \ i = 3 \rightarrow Q_3|q_3\rangle \ \ i = 6 \rightarrow Q_6|q_0\rangle$$

Επομένως, αν η συνθήκη «***if (u & (1 << i))***» είναι:

- true (δηλαδή αποτέλεσμα διαφορετικό από το 0) συμπεραίνεται ότι το στοιχείο $u_{6-i} = 1$ και επομένως εφαρμόζεται μία πύλη Z («***qc.z(qr[6-i])***»).
- false (δηλαδή αποτέλεσμα ίσο με το 0) τότε συμπεραίνεται ότι $u_{6-i} = 0$ και εφαρμόζεται η πύλη Identity («***qc.iden(qr[6-i])***»).

Σύμφωνα με τα αποτελέσματα που προέκυψαν, η ακολουθία των πυλών που θα εφαρμοστούν θα είναι Z, Z, Id, Id, Z, Id, Z (όπου Id εννοείται η πύλη Identity) διότι πρέπει να αντιστοιχεί στην συμβολοσειρά u η οποία δίνεται ως 1100101 (όπου $u_i = 0$ πραγματοποιείται μία Identity πύλη και όπου $u_i = 1$ πραγματοποιείται μία Z πύλη). Σχεδιάζοντας το κβαντικό κύκλωμα παρατηρείται η εν λόγω ακολουθία μέσα στο «μαύρο κουτί».

```
In [3]: qc.draw()
```



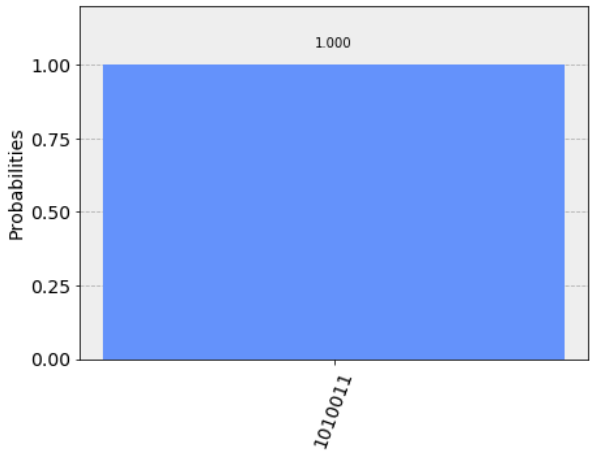
Εικόνα 4-33. Κβαντικό Κύκλωμα Bernstein-Vazirani

Οι κάθετες διακεκομμένες γραμμές πάνω στο γκρι πλαίσιο που παρατηρούνται στο παραπάνω σχήμα αντιπροσωπεύουν τα φράγματα που ορίστηκαν προηγουμένως.

Στην συνέχεια, εκτελώντας το κβαντικό κύκλωμα 1000 φορές και σχεδιάζοντας το ιστόγραμμα των κβαντικών καταστάσεων προκύπτει το εξής αποτέλεσμα:

```
In [4]:
from qiskit.tools.visualization import plot_histogram

backend = Aer.get_backend('qasm_simulator')
job = execute(qc, backend, shots=1000)
result = job.result()
counts = result.get_counts()
plot_histogram(counts)
```



Εικόνα 4-34. Ιστόγραμμα κβαντικών καταστάσεων

Το αποτέλεσμα για όλες τις εκτελέσεις είναι η συμβολοσειρά 1010011. Λαμβάνοντας υπόψη την αναπαράσταση του Qiskit, το οποίο αναπαριστά την κβαντική κατάσταση πρώτα από το τελευταίο qubit ($Q_{n-1} \otimes \ldots \otimes Q_1 \otimes Q_0$), η κατάσταση αυτή αντιστοιχεί στην κβαντική κατάσταση $|1100101\rangle$. Συνεπώς, κατά το τέλος το αλγορίθμου και πραγματοποιώντας τις μετρήσεις σε κάθε qubit, η άγνωστη συμβολοσειρά u μπορεί να ανακτηθεί επιτυχώς στην ολότητά της.

Σε αυτό το σημείο, θα δοκιμαστεί η εκτέλεση του ίδιου κβαντικού κυκλώματος αλλά αυτή την φορά σε έναν πραγματικό κβαντικό υπολογιστή μέσω της IBMQ. Ακολουθεί ο κώδικας για την επιλογή provider της IBMQ και την εύρεση backend με την μικρότερη ουρά αναμονής και με τουλάχιστον 7 qubit.

```
In [5]:
from qiskit import IBMQ

IBMQ.load_account()
provider = IBMQ.get_provider(group='open')

from qiskit.providers.ibmq import least_busy
backend = provider.backends(filters=lambda x: not x.configuration().simulator
                            and x.configuration().n_qubits >= 7)
least_busy(backend)
```

```
<IBMQBackend('ibmq_16_melbourne') from IBMQ(hub='ibm-q', group='open', project='main')>
```

Εικόνα 4-35. Επιλογή provider και εύρεση του λιγότερο απασχολημένου κβαντικού υπολογιστή με 7 qubit

Στην συνέχεια, επιλέγεται ο “ibmq_16_melbourne” ως backend και εκτελείται σε αυτόν το κβαντικό κύκλωμα.

```
In [6]:
backend = provider.get_backend('ibmq_16_melbourne')

from qiskit.tools.monitor import job_monitor

job = execute(qc,backend=backend, shots=1000)
job_monitor(job)

results = job.result()
answer = results.get_counts(qc)
```

```
Job Status: job has successfully run
```

Εικόνα 4-36. Εκτέλεση κβαντικού κυκλώματος σε κβαντικό υπολογιστή

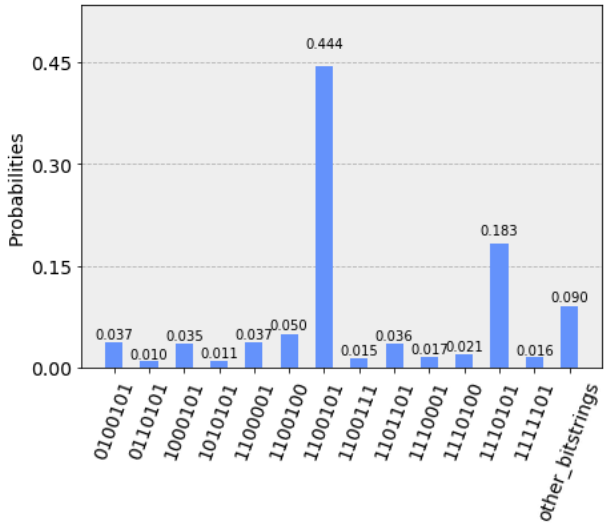
Πριν την εμφάνιση των αποτελεσμάτων σε ένα ιστόγραμμα, για την αποφυγή εμφάνισης πολλών αποτελεσμάτων για λόγους ευ ανάγνωσης, τα αποτελέσματα φιλτράρονται προκειμένου να εμφανίζονται μόνο οι καταστάσεις που εμφανίστηκαν πάνω από το 1% των περιπτώσεων. Όλες οι καταστάσεις οι οποίες εμφανίζονται λιγότερο από το 1% των περιπτώσεων εμφανίζονται συνολικά στο πεδίο “other_bitstrings”.

```
In [7]:
import numpy as np # Import fundamental package for scientific computing

threshold = int(0.01 * 1000) # Threshold of plotting significant measurements
# Filter answer for better view of plots
filteredAnswer = {k: v for k,v in answer.items() if v >= threshold}

removedCounts = np.sum([ v for k,v in answer.items() if v < threshold ]) # Number of counts removed
filteredAnswer['other_bitstrings'] = removedCounts  # Assigne new index for the removed counts

plot_histogram(filteredAnswer)
```



Εικόνα 4-37. Φιλτράρισμα αποτελεσμάτων και ιστόγραμμα κβαντικών καταστάσεων

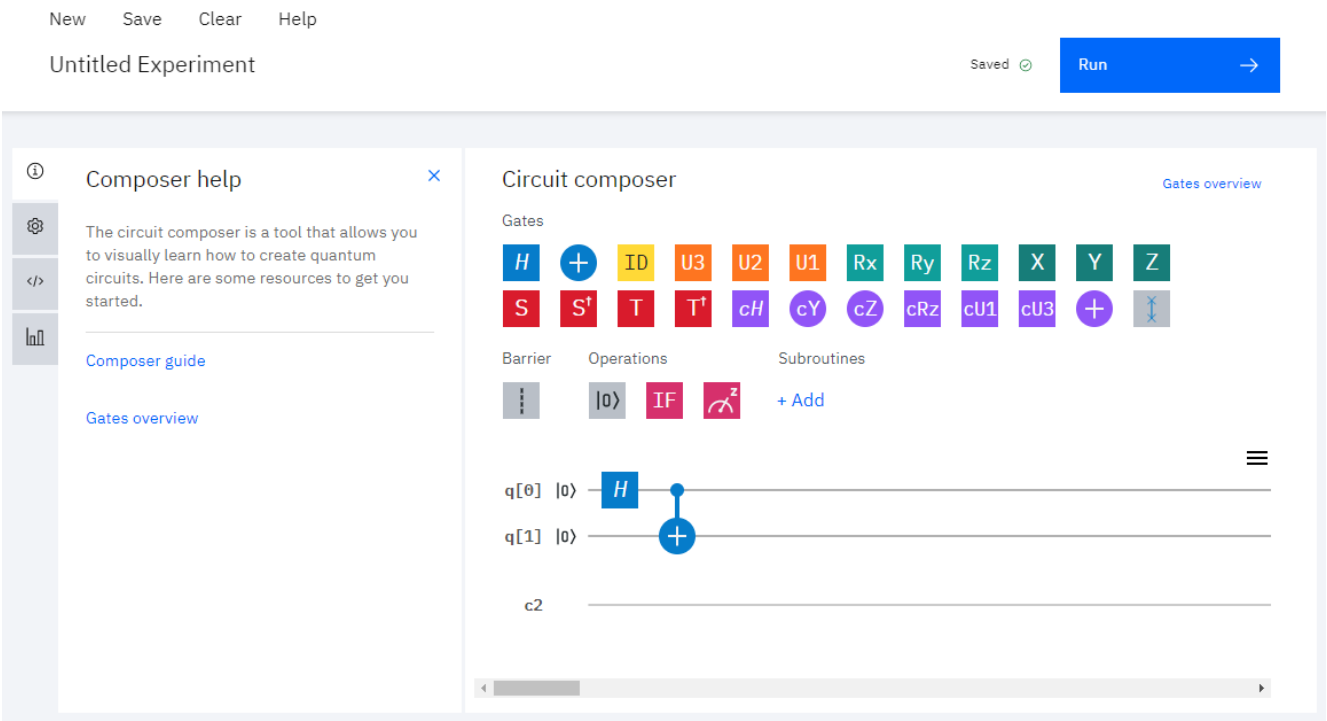
Για άλλη μια φορά, παρατηρούνται τα αναμενόμενα σφάλματα που προκύπτουν από την εκτέλεση ενός κυκλώματος σε μία πραγματική κβαντική υπολογιστική συσκευή. Από την στιγμή όμως που το κύκλωμα είναι σχετικά μικρό είναι δυνατός ο σχηματισμός μίας εύστοχης εκτίμησης. Το επιθυμητό αποτέλεσμα εμφανίζεται με φανερά μεγαλύτερη συχνότητα από ότι τα υπόλοιπα αποτελέσματα, με πιθανότητα εμφάνισης 44.4%. Ανατρέχοντας λίγο πίσω στο πρώτο παράδειγμα που εκτελέστηκε σε κβαντικό υπολογιστή με σκοπό την δημιουργία κβαντικής διεμπλοκής, καθώς το κύκλωμα ήταν αρκετά πιο μικρό, περιέχοντας μόνο δύο qubit και δύο κβαντικές πύλες (Hadamard και CNOT), ως αποτέλεσμα είχε 89.6% ποσοστό ακρίβειας. Γίνεται φανερό πως καθώς τα κβαντικά κυκλώματα μεγαλώνουν, αυξάνεται και ο αριθμός των σφαλμάτων που παρατηρούνται στους κβαντικούς υπολογισμούς.

4.7 IBM Q Experience

Κλείνοντας το κεφάλαιο για το Qiskit, αξίζει να αναφερθούν συνοπτικά δύο ενδιαφέροντα και σημαντικά εργαλεία της πλατφόρμας IBM Q Experience, το **Circuit Composer** και το **Qiskit Notebooks**.

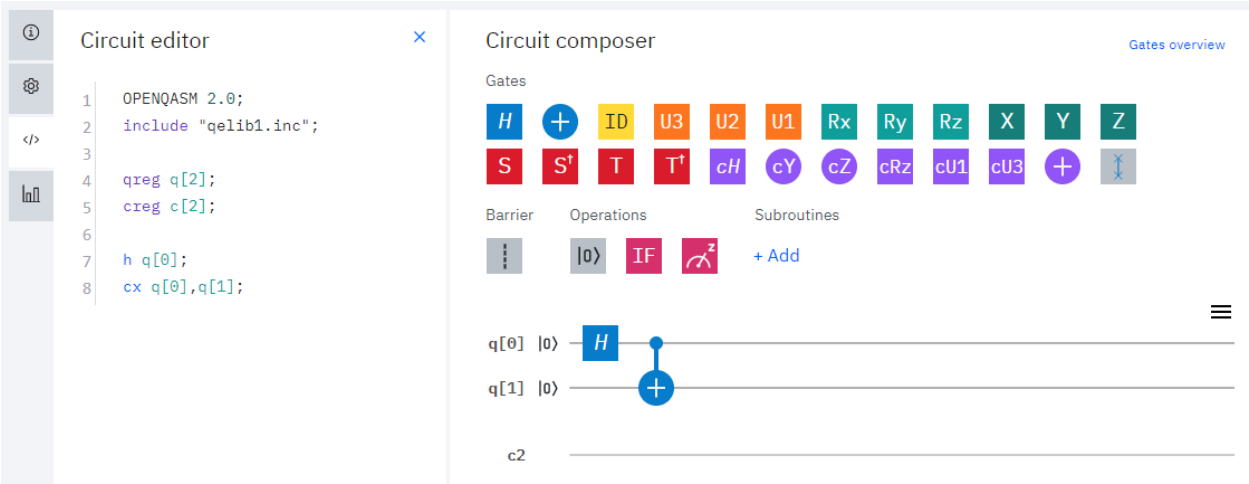
4.7.1 Circuit Composer

Το εργαλείο Circuit Composer δίνει στον χρήστη την δυνατότητα να δημιουργεί τα δικά του κβαντικά κυκλώματα και να τα εκτελεί είτε σε προσομοιωτή είτε σε πραγματικούς κβαντικούς υπολογιστές της IBM Q χωρίς την χρήση κώδικα. Ο χρήστης επιλέγει την πύλη που επιθυμεί και κάνοντας *drag and drop* μπορεί να την τοποθετήσει στο κβαντικό του κύκλωμα. Στο πεδίο «*Gates overview*» παρέχεται μία συνοπτική περιγραφή όλων των πυλών και των λειτουργιών τους.



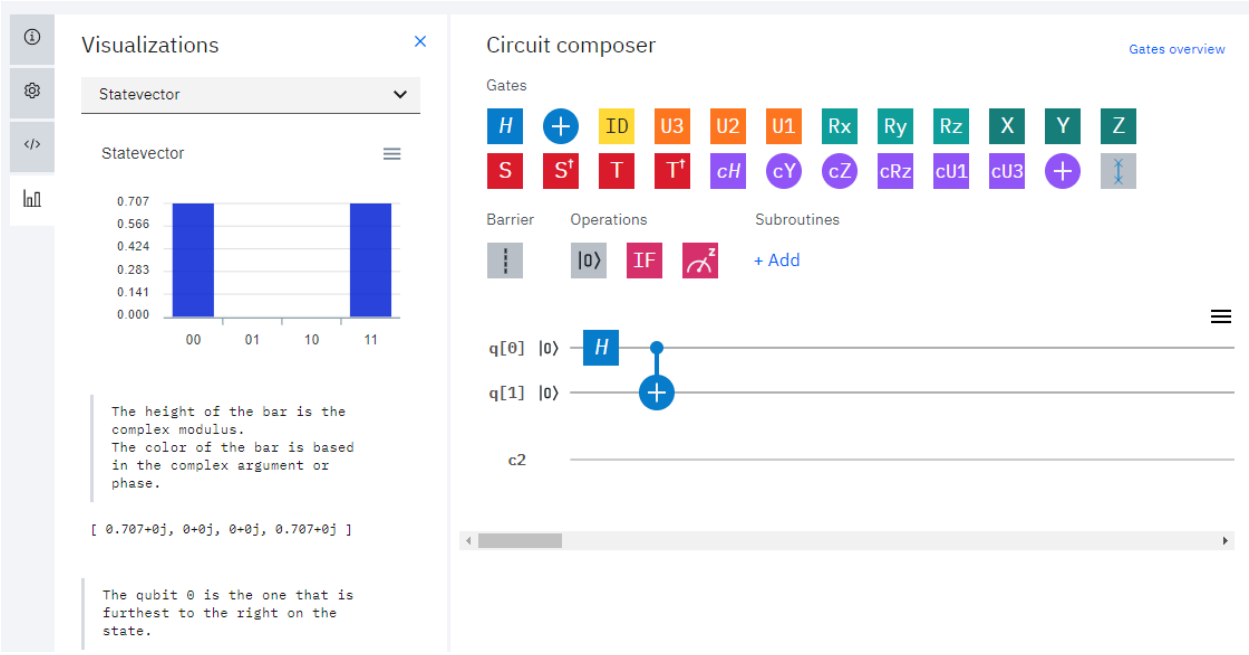
Εικόνα 4-38. Circuit Composer

Παρέχεται επίσης στον χρήστη ο κώδικας OpenQASM, ο οποίος ανανεώνεται κάθε φορά που ο χρήστης πραγματοποιεί μια αλλαγή στο κύκλωμα.



Εικόνα 4-39. Εμφάνιση κώδικα OpenQASM

Επιπλέον, ο χρήστης έχει πρόσβαση σε διάφορες οπτικοποιήσεις (όπως ιστογράμματα, πίνακες πυκνότητας και πιθανότητες εμφάνισης) για τα αναμενόμενα αποτελέσματα του κυκλώματος. Οι οπτικοποιήσεις ανανεώνονται κάθε φορά που ο χρήστης πραγματοποιεί μία αλλαγή στο κύκλωμά του.

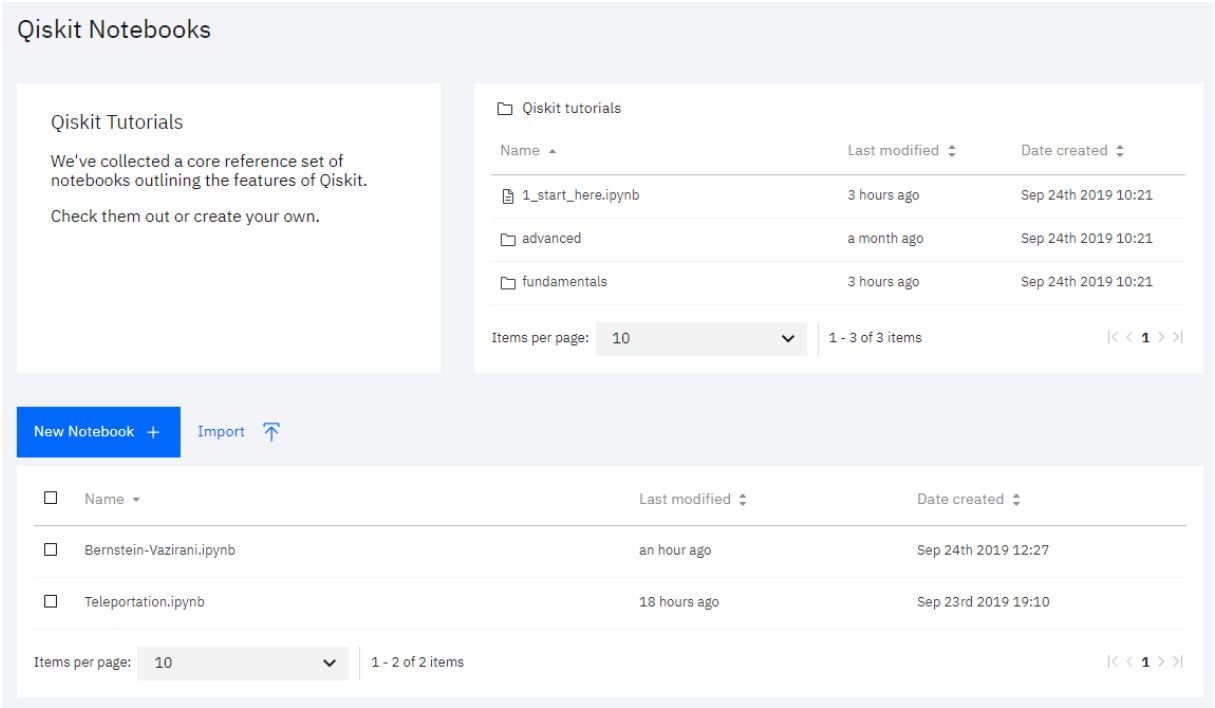


Εικόνα 4-40. Οπτικοποίηση αναμενόμενων αποτελεσμάτων

Τέλος, όταν το κύκλωμα ολοκληρωθεί, ο χρήστης πατώντας το κουμπί «*Run*» εκτελεί το κύκλωμα του επιλέγοντας το backend που επιθυμεί καθώς και τον αριθμό των επαναλήψεων (1, 1024, 4096, ή 8192 επαναλήψεις). Όταν πραγματοποιηθεί η ολοκλήρωση της εκτέλεσης, ο χρήστης ενημερώνεται με σχετικό μήνυμα και μπορεί να ελέγξει τα αποτελέσματα του κυκλώματός του.

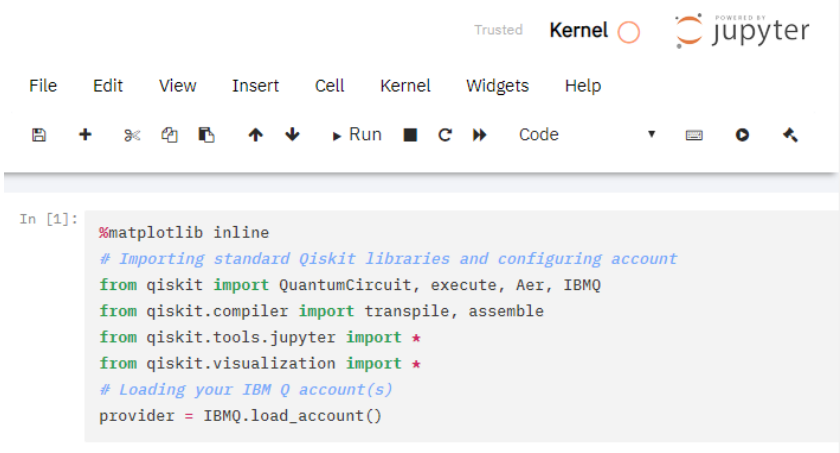
4.7.2 Qiskit Notebooks

Η IBM Q Experience παρέχει επιπλέον το εργαλείο Qiskit Notebooks το οποίο επιτρέπει στον χρήστη να δημιουργεί τα δικά του Jupyter Notebooks και να τα εκτελεί είτε σε προσομοιωτές είτε σε πραγματικούς κβαντικούς υπολογιστές καθώς επίσης παρέχει πρόσβαση στον χρήστη στα tutorials του Qiskit.



Εικόνα 4-41. Qiskit Notebooks

Πατώντας το κουμπί «*New Notebook +*» δημιουργείται αυτόματα ένα αρχείο Notebook αρχικοποιημένο με ορισμένα από τα βασικά imports.



Εικόνα 4-42. Δημιουργία νέου Notebook

Τα παραδείγματα και οι αλγόριθμοι που παρουσιάστηκαν προηγουμένως δημιουργήθηκαν μέσω του Qiskit Notebooks.

5 Quantum Development Kit

5.1 Εισαγωγικά

Το **Microsoft Quantum Development Kit (QDK)** κυκλοφόρησε στις 11 Δεκεμβρίου του 2017 και αποτελεί ένα κιτ ανάπτυξης κβαντικού λογισμικού. Παρέχει ένα πλήρες περιβάλλον για την ανάπτυξη και την προσομοίωση κβαντικών κυκλωμάτων. Υποστηρίζεται στις πλατφόρμες Windows, macOS και Linux. Ο κβαντικός προσομοιωτής που περιέχεται στο Quantum Development Kit είναι ικανός να επεξεργαστεί μέχρι και 32 qubit σε έναν τοπικό υπολογιστή και έως και 40 qubit στην cloud πλατφόρμα Azure της Microsoft. Τα κβαντικά προγράμματα μπορούν να υλοποιηθούν και να τρέξουν στο **Visual Studio** καθώς και στο **VSCode**.

Εικόνα 5-1. Microsoft Quantum Development Kit

5.2 Γλώσσα Προγραμματισμού Q#

Το **Quantum Development Kit** χρησιμοποιεί την γλώσσα προγραμματισμού **Q#** η οποία κυκλοφόρησε από την Microsoft ως μέρος του συγκεκριμένου κιτ. Η γλώσσα προγραμματισμού Q# είναι μία domain-specific γλώσσα η οποία χρησιμοποιείται για την υλοποίηση κβαντικών αλγορίθμων. Η Q# διατίθεται ξεχωριστά ως επέκταση για το Visual Studio αλλά μπορεί επίσης να εκτελεστεί και ως ανεξάρτητο εργαλείο από την Γραμμή Εντολών ή/και το VSCode. Ο κβαντικός προσομοιωτής που παρέχει το Quantum Development Kit είναι ικανός να εκτελέσει την Q#. Για την επίκληση του κβαντικού προσομοιωτή χρησιμοποιείται μία άλλη γλώσσα προγραμματισμού .NET (συνήθως C#), η οποία παρέχει τα κλασσικά δεδομένα εισόδου για τον προσομοιωτή και διαβάζει τα κλασσικά δεδομένα εξόδου από τον προσομοιωτή.

5.3 Quantum Katas

Η Microsoft παρέχει ένα σύνολο από self-paced tutorials, γνωστά ως **Quantum Katas**, διαθέσιμα στο GitHub τα οποία βοηθούν στην εκμάθηση της γλώσσας Q#. Τα tutorials αυτά είναι διαθέσιμα ως Jupyter Notebooks αλλά μπορούν να εκτελεστούν και τοπικά μέσω του Visual Studio ή του VSCode.

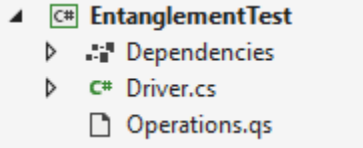
Τα tutorials είναι χωρισμένα σε θεματικές ενότητες. Κάθε θεματική αποτελείται από μία σειρά από εργασίες. Για κάθε μία από τις εργασίες ο χρήστης καλείται να συμπληρώσει τον κώδικα ο οποίος λείπει. Καθώς οι εργασίες προχωράνε, αυξάνεται και το επίπεδο δυσκολίας τους. Παράλληλα παρέχονται στον χρήστη ορισμένα hints και πληροφορίες που ίσως χρειαστεί για την επίλυση των εργασιών. Όταν ο χρήστης συμπληρώσει τις απαντήσεις του, μπορεί στη συνέχεια να τις ελέγξει. Σε περίπτωση που ο χρήστης δεν βρίσκει την απάντηση για μία εργασία, μπορεί να ανατρέξει στις λύσεις των προβλημάτων.

5.4 Μοντέλο προγραμματισμού Quantum Development Kit

Στο μοντέλο προγραμματισμού κβαντικών κυκλωμάτων του Quantum Development Kit ο κβαντικός υπολογιστής αντιμετωπίζεται ως **συνεπεξεργαστής (coprocessor)**. Οι λογικές πράξεις εκτελούνται σε ένα κλασσικό «host» υπολογιστή και μόνο όταν είναι απαραίτητο, το «host» πρόγραμμα μπορεί να επικαλεστεί μία υπορουτίνα που εκτελείται στον συνεπεξεργαστή. Όταν η υπορουτίνα ολοκληρωθεί τότε το «host» πρόγραμμα μπορεί να έχει πρόσβαση στα αποτελέσματά της.

Η Q# χρησιμοποιείται για τη σύνταξη υπορουτινών οι οποίες εκτελούνται σε έναν συνεπεξεργαστή, υπό τον έλεγχο ενός κλασσικού προγράμματος και υπολογιστή. Οι υπορουτίνες Q# προς το παρόν εκτελούνται μόνο σε προσομοιωτή.

Δημιουργώντας ένα *Q# application* στο Visual Studio, παράγονται δύο αρχεία, το **Driver.cs** και το **Operation.qs**. Το αρχείο ".cs" αντιστοιχεί στο κλασσικό «host» πρόγραμμα, σε γλώσσα C#, στο οποίο πραγματοποιούνται οι λογικές πράξεις και το αρχείο ".qs" αντιστοιχεί στο πρόγραμμα που εκτελείται στον συνεπεξεργαστή, σε γλώσσα Q#.



Εικόνα 5-2. Δημιουργία Q# application

5.5 Βασικοί Τύποι Μεταβλητών της Q#

Ορισμένοι βασικοί τύποι μεταβλητών που χρησιμοποιούνται στην Q# είναι οι εξής:

- **Int:** αντιπροσωπεύει έναν ακέραιο αριθμό.
- **Double:** αντιπροσωπεύει ένα δεκαδικό αριθμό.
- **Bool:** αντιπροσωπεύει μία Boolean μεταβλητή.
- **Qubit:** αντιπροσωπεύει ένα qubit. Ο τύπος μεταβλητής *Qubit* αποτελεί ένα **opaque data type**, δηλαδή ο χρήστης δεν έχει άμεση πρόσβαση στο περιεχόμενο της μεταβλητής. Οι ενέργειες πάνω σε *Qubit* υλοποιούνται με την χρήση εντολών οι οποίες εκτελούνται σε ένα κβαντικό επεξεργαστή (ή κβαντικό προσομοιωτή σε αυτή την περίπτωση).

- **Result:** αντιπροσωπεύει το αποτέλεσμα μίας μέτρησης. Δέχεται δύο πιθανές τιμές, «Zero» για την κατάσταση $|0\rangle$ και «One» για την κατάσταση $|1\rangle$.
- **Unit:** επιτρέπει μόνο την τιμή «()». Αυτός ο τύπος χρησιμοποιείται για να υποδείξει ότι μία μέθοδος Q# δεν επιστρέφει κάποια τιμή.

5.6 Διανομή των Qubit

Τα qubit στην Q# διανέμονται στον χρήστη εντός του μπλοκ «***using***». Κατά την διανομή των qubit στον χρήστη, τα qubit αρχικοποιούνται στην κατάσταση $|0\rangle$.

```
using (register = Qubit[3]){
    // Do stuff...
```

Εικόνα 5-3. Διανομή qubit

Σύμφωνα με το παραπάνω παράδειγμα, ο χρήστης δημιουργεί τρεις μεταβλητές *Qubit* (δεσμεύει δηλαδή τρία qubit) αρχικοποιημένες στην κατάσταση $|000\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle$ τις οποίες αποθηκεύει στην μεταβλητή *register*.

Κατά την αποδέσμευση των qubit, αναμένεται όλα τα qubit που χρησιμοποιήθηκαν να επιστρέψουν στην κατάσταση $|0\rangle$, έτσι ώστε να είναι δυνατή η επαναχρησιμοποίησή τους. Η Q# προσφέρει την λειτουργία «***Reset***» η οποία φροντίζει το qubit να επιστρέψει στην κατάσταση $|0\rangle$, πραγματοποιώντας μία μέτρηση στο qubit και ενεργώντας κατάλληλα. Μια τέτοια μέτρηση θα καταστρέψει κάθε πιθανή διεμπλοκή με τα υπόλοιπα qubit η οποία θα μπορούσε να επηρεάσει τους μετέπειτα υπολογισμούς. Πέρα από την λειτουργία «*Reset*», η οποία δέχεται ως είσοδο ένα μόνο *Qubit* υπάρχει και η λειτουργία «***ResetAll***» η οποία δέχεται ως είσοδο έναν πίνακα από *Qubit* και χρησιμοποιείται συνήθως για να επαναφέρει όλα τα δεσμευμένα qubit στην κατάσταση $|0\rangle$.

5.7 Mutable και Immutable μεταβλητές

Οι μεταβλητές στην Q# διακρίνονται σε δύο βασικές κατηγορίες, ***immutable*** και ***mutable***.

Οι *immutable* μεταβλητές αρχικοποιούνται με το keyword «***let***». Οι μεταβλητές αυτές από την στιγμή που έχουν αρχικοποιηθεί δεν μπορούν να αλλάξουν τιμή. Ένα ***immutable binding*** αποτελείται από το keyword «*let*», το όνομα της μεταβλητής, το σύμβολο "=" και ένα *expression* που αντιστοιχεί στο όνομα της μεταβλητής που ορίστηκε, ακολουθούμενο από τον *χαρακτήρα τερματισμού πρότασης* ";".

```
using (register = Qubit[1]){

    let qubit0 = register[0];
```

Εικόνα 5-4. Παράδειγμα immutable binding

Οι *mutable* μεταβλητές, σε αντίθεση με τις *immutable* μεταβλητές, μπορούν να αλλάξουν την τιμή τους ακόμη και μετά την αρχικοποίησή τους. Η αρχικοποίηση μια *mutable* μεταβλητής αποτελείται από την λέξη «*mutable*», το όνομα της μεταβλητής, το σύμβολο "=" και ένα *expression* που αντιστοιχεί στο όνομα της μεταβλητής που ορίστηκε, ακολουθούμενο από τον *χαρακτήρα τερματισμού πρότασης* ";". Η αλλαγή της τιμής μιας *mutable* μεταβλητής πραγματοποιείται με την χρήση του keyword «***set***». Ο τύπος μεταβλητής που επιλέγεται, συμπεραίνεται από το *expression* που ορίστηκε. Αν η τιμή της μεταβλητής αλλάξει στην πορεία, η αλλαγμένη τιμή πρέπει να είναι συμβατή με τον αρχικό τύπο μεταβλητής που ορίστηκε. Στο παρακάτω παράδειγμα, συμπεραίνεται ότι ο τύπος μεταβλητής που ορίστηκε είναι *integer*.

```
mutable counts = 0;
for (i in 1 .. 2 .. 10) {
    set counts += 1;
}
```

Εικόνα 5-5. Παράδειγμα mutable binding

5.8 Δημιουργία Κβαντικής Διεμπλοκής

Για την δημιουργία κβαντικής διεμπλοκής δημιουργείται ένα *Q# Application* με όνομα "*EntanglementTest*" και δημιουργούνται αυτόματα τα αρχεία *Operation.qs* και *Driver.cs*.

Στο αρχείο *Operation.qs* εκτελείται η υπορουτίνα για την δημιουργία κβαντικής διεμπλοκής. Στο αρχείο αυτό εφαρμόζονται όλες οι κβαντικές πύλες οι οποίες στην συνέχεια εκτελούνται στον κβαντικό προσομοιωτή. Ακολουθεί ο κώδικας Q# για την δημιουργία κβαντικής διεμπλοκής μεταξύ δύο qubit.

```
1   namespace Quantum.EntanglementTest
2   {
3       open Microsoft.Quantum.Intrinsic;
4       open Microsoft.Quantum.Canon;
5
6       operation Entangle () : (Result,Result) {
7
8           mutable result = (Zero,Zero);
9
10          // The following using block creates two qubits and initializes them in |00⟩
11          using (register = Qubit[2]){


14              let qubit0 = register[0];
15              let qubit1 = register[1];
16
17              // Apply Hadamard and  CNOT to create the Bell state 1/sqrt(2)(|00⟩+|11⟩)
18              H(qubit0);
19              CNOT(qubit0,qubit1);
20
21              // Measure qubits
22              set result = (M(qubit0),M(qubit1));
23
24              // Reset all qubits before releasing them
25              ResetAll(register);
26          }
27          return result;
28      }
29  }
```

Εικόνα 5-6. Operation.qs

Αρχικά, στην γραμμή 6 του κώδικα δημιουργείται η υπορουτίνα με όνομα "*Entangle*" μέσα στην οποία θα περιληφθεί ο κώδικας για την κβαντική διεμπλοκή δύο qubit. Ως έξοδο, η υπορουτίνα επιστρέφει δύο μεταβλητές τύπου *Result* οι οποίες θα αντιστοιχούν στην κατάσταση του κάθε qubit μετά την μέτρηση της κατάστασης Bell $|\Phi^+\rangle$.

Στην γραμμή 8 δημιουργείται η *mutable* μεταβλητή με όνομα "*result*", αρχικοποιημένη με τις τιμές *Zero* και *Zero*, στην οποία θα αποθηκευτούν τα αποτελέσματα της μέτρησης της κατάστασης Bell $|\Phi^+\rangle$. Ο τύπος της μεταβλητής "*result*" συμπεραίνεται ότι είναι τύπος *Result*.

Στην γραμμή 11 δεσμεύονται δύο qubit, αρχικοποιημένα στην κατάσταση $|00\rangle$, στην μεταβλητή με όνομα "*register*". Στην συνέχεια, στις γραμμές 14 και 15, τα περιεχόμενα της μεταβλητής "register" γίνονται *bind* στις *immutable* μεταβλητές "*qubit0*" και "*qubit1*" με την χρήση του keyword «*let*».

Στην γραμμή 18 εφαρμόζεται μία πύλη Hadamard στο πρώτο qubit και στην γραμμή 19 εφαρμόζεται μία πύλη CNOT με το "*qubit0*" ως control qubit και το "*qubit1*" ως target qubit. Στο σημείο αυτό του αλγορίθμου η κβαντική κατάσταση του συστήματος είναι η κατάσταση $|\Phi^+\rangle$.

Στην γραμμή 22 πραγματοποιούνται δύο μετρήσεις στα δύο qubit και τα αποτελέσματα των μετρήσεων αποθηκεύονται στην *mutable* μεταβλητή "*result*" με την χρήση του keyword «*set*».

Στην γραμμή 25 με την χρήση του «***ResetAll(register);***», τα qubit που χρησιμοποιήθηκαν επιστρέφουν στην κατάσταση $|00\rangle$ ώστε να είναι δυνατή η επαναχρησιμοποίησή τους.

Τέλος, στην γραμμή 27 επιστρέφονται τα αποτελέσματα της μέτρησης.

Έχοντας ολοκληρώσει την κατασκευή της υπορουτίνας σε Q#, ακολουθεί η δημιουργία του κλασσικού προγράμματος σε C# το οποίο θα καλέσει την υπορουτίνα που μόλις ολοκληρώθηκε και θα εξετάσει τα αποτελέσματα. Ακολουθεί ο κώδικας C#.

```
1   using System;
2   using System.Linq;
3   using Microsoft.Quantum.Simulation.Core;
4   using Microsoft.Quantum.Simulation.Simulators;
5
6   namespace Quantum.EntanglementTest
7   {
        0 references
8       class Driver
9       {
            0 references
10          static void Main(string[] args)
11          {
12              System.Console.WriteLine($"");
13              // Run the quantum code on a "co-processor"
14              using (var qsim = new QuantumSimulator())
15              {
16                  foreach (var i in Enumerable.Range(0, 10))
17                  {
18                      // Simulate operation Entangle
19                      var results = Entangle.Run(qsim).Result;
20                      // Print results
21                      System.Console.WriteLine($" Measured {results}.");
22                  }
23                  System.Console.WriteLine("\n\n Press any key to exit...\n\n");
24                  System.Console.ReadLine();
25              }
26          }
27      }
28  }
```

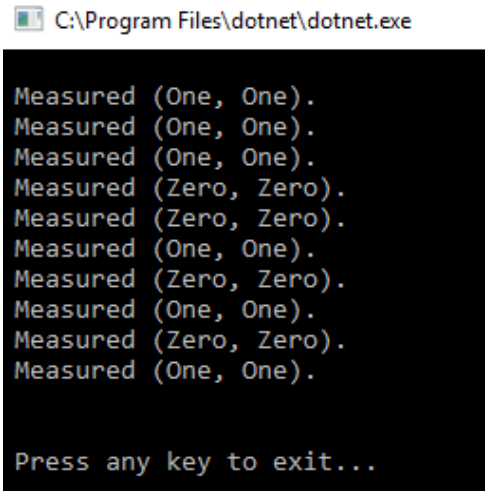
Εικόνα 5-7. Driver.cs

Στην γραμμή 14 δημιουργείται ένα *instance* του κβαντικού προσομοιωτή του Quantum Development Kit στην μεταβλητή "*qsim*". Η μεταβλητή αυτή στη συνέχεια θα χρησιμοποιείται για την προσομοίωση των διάφορων υπορουτινών που δημιουργούνται στον κβαντικό κώδικα Q#.

Στις γραμμές 16-22 εκτελείται η υπορουτίνα "*Entangle*" στον κβαντικό προσομοιωτή 10 φορές.

Πιο συγκεκριμένα, στην γραμμή 19 η μεταβλητή "*qsim*", που αντιστοιχεί στον κβαντικό προσομοιωτή, χρησιμοποιείται για την προσομοίωση («***.Run()***») της υπορουτίνας "*Entangle*", η οποία δημιουργήθηκε στο αρχείο *Operation.qs*, προκειμένου να διαβάσει τα αποτελέσματά της («***.Result()***»). Τα αποτελέσματα της υπορουτίνας αποθηκεύονται στην μεταβλητή "*results*".

Στην γραμμή 21 τυπώνονται τα αποτελέσματα που μετρήθηκαν. Τρέχοντας το πρόγραμμα προκύπτουν τα εξής αποτελέσματα:



Εικόνα 5-8. Αποτελέσματα

Αν και το δείγμα είναι πολύ μικρό με μόνο δέκα επαναλήψεις, παρατηρούνται τα αναμενόμενα αποτελέσματα, δηλαδή περίπου τις μισές φορές το σύστημα βρίσκεται στην κατάσταση $|00\rangle$ και τις υπόλοιπες στην κατάσταση $|11\rangle$. Στο συγκεκριμένο παράδειγμα παρατηρήθηκε η κατάσταση $|00\rangle$ έξι φορές και η κατάσταση $|11\rangle$ τέσσερις φορές.

5.9 Κβαντική Τηλεμεταφορά

Στο κεφάλαιο αυτό θα αναλυθεί ο αλγόριθμος της κβαντικής τηλεμεταφοράς σύμφωνα με το κβαντικό μοντέλο προγραμματισμού του Quantum Development Kit. Στο παράδειγμα αυτό, θα επιλέγεται τυχαία μία κβαντική κατάσταση ($|0\rangle$ ή $|1\rangle$) από το κλασσικό πρόγραμμα η οποία στη συνέχεια θα τηλεμεταφέρεται από το q_{trans} στο q_{bob}. Η υλοποίηση της κβαντικής τηλεμεταφοράς πραγματοποιείται στην υπορουτίνα "*Teleport*" σε Q# στο αρχείο *Operation.qs*.

```
6   operation Teleport (message: Bool) : Bool {
7
8       mutable result = false;
9
10      using (register = Qubit[3])
11      {
12
13          let qtrans = register[0];
14          let qalice = register[1];
15          let qbob = register[2];
16
17          // Set qtrans
18          // Encode the message we want to send
19          if (message) {
20              X(qtrans);
21          }
22
23          // Entangle qalice and qbob
24          H(qalice);
25          CNOT(qalice,qbob);
```

Εικόνα 5-9. Αρχικοποίηση qtrans και κβαντική διεμπλοκή μεταξύ qalice και qbob

Στην γραμμή 6 η υπορουτίνα "*Teleport*" δέχεται ως είσοδο μία *Boolean* μεταβλητή με το όνομα "*message*" της οποίας η τιμή θα αντιστοιχεί στην κατάσταση του qubit q_{trans} που πρόκειται να τηλεμεταφερθεί στο q_{bob}. Η μεταβλητή "*message*" επομένως δέχεται δύο πιθανές τιμές true και false οι οποίες αντιστοιχούν στις καταστάσεις $|1\rangle$ και $|0\rangle$. Ως έξοδο, επιστρέφει μία *Boolean* μεταβλητή η οποία θα αντιπροσωπεύει την κατάσταση του q_{bob} μετά την μέτρηση του, κατά το τέλος του αλγορίθμου.

Στην γραμμή 8 δημιουργείται η mutable μεταβλητή με όνομα "*result*", αρχικοποιημένη με την τιμή false, η οποία αντιστοιχεί στην κατάσταση $|0\rangle$, στην οποία θα αποθηκευτεί το αποτέλεσμα της μέτρησης του q_{bob} κατά το πέρας του αλγορίθμου.

Στην γραμμή 11 δεσμεύονται τρία qubit, αρχικοποιημένα στην κατάσταση $|000\rangle$, στην μεταβλητή με όνομα "*register*". Στην συνέχεια, στις γραμμές 13, 14 και 15 τα περιεχόμενα της μεταβλητής "*register*" γίνονται *bind* στις *immutable* μεταβλητές "*qtrans*" και "*qalice*" και "*qbob*".

Στις γραμμές 19-21 η κατάσταση του q_{trans} ορίζεται σύμφωνα με την τιμή της μεταβλητής "*message*".

- Αν το «*message = true*» τότε το qtrans πρέπει να οριστεί στην κατάσταση $|1\rangle$.
- Αν το «*message = false*» τότε το qtrans πρέπει να οριστεί στην κατάσταση $|0\rangle$.

Επομένως, αν το «*message = true*» τότε εφαρμόζεται μία πύλη X στο qubit q_{trans}. Αν το «*message = false*» τότε δεν πραγματοποιείται κάποια ενέργεια διότι το q_{trans} είναι ήδη αρχικοποιημένο στην κατάσταση $|0\rangle$.

Στις γραμμές 24-25 εφαρμόζεται μία πύλη Hadamard στο q_{alice} και έπειτα εφαρμόζεται μία πύλη CNOT με το q_{alice} ως control qubit και το q_{bob} ως target qubit.

```
26
27              // Step 1 - Apply CNOT to qtrans and qalice
28              CNOT(qtrans,qalice);
29              // Step 2 - Apply Hadamard to qtrans
30              H(qtrans);
31
32              // Step 3
33              // Measure qtrans
34              // If qtrans=|1⟩, apply Z to qbob
35              if (M(qtrans) == One) {
36                  Z(qbob);
37              }
38              // Step 4
39              // Measure qalice
40              // If qalice=|1⟩, apply X to qbob
41              if (M(qalice) == One) {
42                  X(qbob);
43              }
44
45              // Measure qbob
46              if (M(qbob) == One) {
47                  set result = true;
48              }
49              ResetAll(register);
50
51              return result;
52          }
53      } // end operation Teleport
```

Εικόνα 5-10. Αλγόριθμος κβαντικής τηλεμεταφοράς

Στην γραμμή 28 εφαρμόζεται μία πύλη CNOT με το q_{trans} ως control qubit και το q_{alice} ως target qubit.

Στην γραμμή 30 εφαρμόζεται μία πύλη Hadamard στο q_{trans}.

Στις γραμμές 35-37 πραγματοποιείται μία μέτρηση στο q_{trans}. Αν το q_{trans} βρεθεί στην κατάσταση $|1\rangle$ εφαρμόζεται μία πύλη Z στο q_{bob}. Η συνθήκη «***M(qtrans) == One***» είναι true όταν το αποτέλεσμα της μέτρησης του q_{trans} είναι **One** (κατάσταση $|1\rangle$).

Στις γραμμές 41-43 πραγματοποιείται μία μέτρηση στο q_{alice}. Αν το q_{alice} βρεθεί στην κατάσταση $|1\rangle$ εφαρμόζεται μία πύλη X στο q_{bob}. Η συνθήκη «***M(qalice) == One***» είναι true όταν το αποτέλεσμα της μέτρησης του q_{alice} είναι **One** (κατάσταση $|1\rangle$).

Στις γραμμές 46-48 πραγματοποιείται μία μέτρηση στο q_{bob}, το αποτέλεσμα της οποίας αποθηκεύεται στην μεταβλητή "*result*". Αν το αποτέλεσμα της μέτρησης είναι η κατάσταση $|1\rangle$ τότε η τιμή της μεταβλητής "*result*" ορίζεται σε true. Διαφορετικά, αν το αποτέλεσμα της μέτρησης είναι η κατάσταση

$|0\rangle$ τότε δεν πραγματοποιείται κάποια αλλαγή στην μεταβλητή "*result*" διότι είναι ήδη αρχικοποιημένη ως false η οποία αντιστοιχεί στην κατάσταση $|0\rangle$.

Στην γραμμή 49 τα qubit που χρησιμοποιήθηκαν επιστρέφουν στην κατάσταση $|000\rangle$ ώστε να είναι δυνατή η επαναχρησιμοποίησή τους.

Τέλος, στην γραμμή 51 επιστρέφεται το αποτέλεσμα του αλγορίθμου.

Έχοντας ολοκληρώσει την κατασκευή της υπορουτίνας σε Q#, ακολουθεί η δημιουργία του κλασσικού προγράμματος σε C# το οποίο θα καλέσει την υπορουτίνα "*Teleport*" και θα εξετάσει τα αποτελέσματα. Ακολουθεί ο κώδικας C#.

```
10      static void Main(string[] args)
11      {
12          System.Console.WriteLine();
13          using (var qsim = new QuantumSimulator())
14          {
15              Random rand = new Random();
16
17              foreach (var idxRun in Enumerable.Range(1, 10))
18              {
19                  int sent = rand.Next(2); // Select 0 or 1 at random
20                  // Simulate operation Teleport
21                  var received = Teleport.Run(qsim, Convert.ToBoolean(sent)).Result;
22                  // Convert result to Int
23                  int receivedInt = received ? 1 : 0;
24                  // Print Results
25                  System.Console.Write($"  Test {idxRun}:\tSent {sent},\tgot {receivedInt}.");
26                  System.Console.WriteLine(Convert.ToBoolean(sent) == received ?
27                      "  Teleportation successful.\n" : "\n");
28              }
29          }
30          System.Console.WriteLine("\n\n Press Enter to continue...\n\n");
31          System.Console.ReadLine();
32      }
```

Εικόνα 5-11. Driver.cs

Στην γραμμή 13 δημιουργείται ένα instance του κβαντικού προσομοιωτή του Quantum Development Kit στην μεταβλητή "*qsim*".

Στις γραμμές 17-29 εκτελείται η υπορουτίνα "*Teleport*" στον κβαντικό προσομοιωτή 10 φορές. Για κάθε μία από τις δέκα εκτελέσεις επιλέγεται κάθε φορά τυχαία (γραμμή 19) ο ακέραιος αριθμός 0 ή 1 (οι οποίοι αντιστοιχούν στις καταστάσεις $|0\rangle$ και $|1\rangle$) ο οποίος θα σταλεί ως είσοδο στην υπορουτίνα "*Teleport*" και αποθηκεύεται στην μεταβλητή με όνομα "*sent*".

Πιο συγκεκριμένα, στην γραμμή 21 η μεταβλητή "*qsim*", χρησιμοποιείται για την προσομοίωση της υπορουτίνας "*Teleport*" μαζί με την μεταβλητή "*sent*", η οποία μετατρέπεται σε *Boolean* προκειμένου να είναι συμβατή με την είσοδο της υπορουτίνας. Τα αποτελέσματα της υπορουτίνας αποθηκεύονται στην μεταβλητή "*received*".

Στην γραμμή 23 το αποτέλεσμα της μεταβλητής "*received*" αποθηκεύεται στην μεταβλητή "*receivedInt*" ως integer προκειμένου να συγκριθεί με την μεταβλητή "*sent*" και να προσδιοριστεί αν η κβαντική τηλεμεταφορά πραγματοποιήθηκε επιτυχώς.

Στις γραμμές 25-27 τυπώνονται τα αποτελέσματα του αλγορίθμου και αν η τηλεμεταφορά πραγματοποιήθηκε επιτυχώς τυπώνεται το μήνυμα “Teleportation successful.”. Εκτελώντας το πρόγραμμα, προκύπτουν τα εξής αποτελέσματα:

```
C:\Program Files\dotnet\dotnet.exe

Test 1:       Sent 0, got 0.  Teleportation successful.
Test 2:       Sent 0, got 0.  Teleportation successful.
Test 3:       Sent 1, got 1.  Teleportation successful.
Test 4:       Sent 1, got 1.  Teleportation successful.
Test 5:       Sent 1, got 1.  Teleportation successful.
Test 6:       Sent 0, got 0.  Teleportation successful.
Test 7:       Sent 1, got 1.  Teleportation successful.
Test 8:       Sent 0, got 0.  Teleportation successful.
Test 9:       Sent 0, got 0.  Teleportation successful.
Test 10:      Sent 1, got 1.  Teleportation successful.

Press Enter to continue...
```

Εικόνα 5-12. Αποτελέσματα αλγορίθμου κβαντικής τηλεμεταφοράς

Η κβαντική τηλεμεταφορά πραγματοποιήθηκε επιτυχώς καθ’ όλες τις εκτελέσεις του προγράμματος.

5.10 Αλγόριθμός Bernstein-Vazirani

Στο κεφάλαιο αυτό θα αναλυθεί ο αλγόριθμος Bernstein-Vazirani σύμφωνα με το κβαντικό μοντέλο προγραμματισμού του Quantum Development Kit. Η υλοποίηση των κβαντικών υπολογισμών πραγματοποιείται στις υπορουτίνες “*Bernstein_Vazirani*” και “*Oracle*” σε Q#. Η συμβολοσειρά *u* που θα χρησιμοποιηθεί στο συγκεκριμένο παράδειγμα θα είναι μεγέθους 7 bit.

```
6   operation Oracle (u: Int[], q: Qubit[]): Unit{
7
8       for (i in 0 .. 6) {
9               if(u[i]==1){
10                  Z(q[i]);
11              }
12      }
13  }
```

Εικόνα 5-13. Υπορουτίνα "Oracle"

Η υπορουτίνα "*Oracle*" εκτελεί το «μαύρο κουτί» στο οποίο υλοποιείται ο μετασχηματισμός Q_f. Υπενθυμίζεται ότι, ο μετασχηματισμός Q_f αποσυντίθεται σε n-κβαντικές πύλες, όπου η κάθε μία δρα στο qubit που της αντιστοιχεί με τον εξής τρόπο:

$$Q_i = \{I \text{ , αν } u_i = 0 \; Z \text{ , αν } u_i = 1$$

Δέχεται ως είσοδο έναν πίνακα μεταβλητών *Int* με όνομα "*u*", ο οποίος περιλαμβάνει τα στοιχεία της συμβολοσειράς u και ένα πίνακα μεταβλητών *Qubit* με όνομα "*q*". Από την στιγμή που δεν επιστρέφει κάποια τιμή η υπορουτίνα "*Oracle*", ως έξοδο χρησιμοποιείται ο τύπος μεταβλητής *Unit*.

Στις γραμμές 8-12 υλοποιούνται οι n-κβαντικές πύλες Q_i. Από την στιγμή που η άγνωστη συμβολοσειρά που θα επιλεχθεί είναι μεγέθους 7 bit, θα χρειαστούν 7 επαναλήψεις (μία για κάθε qubit). Αν το στοιχείο $u_i = 1$ τότε εφαρμόζεται μία πύλη Z στο αντίστοιχο qubit, διαφορετικά εφαρμόζεται μία πύλη I. Μία πύλη I στην ουσία, δεν πραγματοποιεί κάποια αλλαγή στην κατάσταση του qubit οπότε ισοδύναμα αν $u_i = 0$, δεν πραγματοποιείται κάποια ενέργεια στο αντίστοιχο qubit.

```
15  operation Bernstein_Vazirani (u: Int[]) : Int[] {
16
17      // Create seven qubits and initialize them in |0000000⟩ state
18      using (register = Qubit[7])
19      {
20          mutable result = new Int[7];
21
22          // Step 1 - Apply Hadamard gates before querying the oracle
23          ApplyToEach(H, register);
24          // Step 2 - Apply the oracle
25          Oracle(u,register);
26          // Step 3 - Apply Hadamard gates after querying the oracle
27          ApplyToEach(H, register);
28
29          // Measure and convert each measurement to an Int
30          for (i in 0 .. 6) {
31              if (M(register[i]) == One) {
32                  set result w/= i <- 1;
33              }
34          }
35
36          // Reset all qubits before releasing them
37          ResetAll(register);
38          return result;
39      }
40  }
```

Εικόνα 5-14. Υπορουτίνα "Bernstein_Vazirani"

Η υπορουτίνα "*Bernstein_Vazirani*" εκτελεί τον αλγόριθμο Bernstein-Vazirani. Δέχεται ως είσοδο έναν πίνακα μεταβλητών *Int* με όνομα "*u*", ο οποίος περιλαμβάνει τα στοιχεία της συμβολοσειράς u και επιστρέφει έναν πίνακα μεταβλητών *Int* ο οποίος περιλαμβάνει τα στοιχεία των μετρήσεων.

Στην γραμμή 18 δεσμεύονται εφτά qubit, αρχικοποιημένα στην κατάσταση $|0\rangle$ το καθένα, στην μεταβλητή με όνομα "*register*".

Στην γραμμή 20 ορίζεται η *mutable* μεταβλητή με όνομα "*result*" ως ένας πίνακας μεταβλητών *Int* μεγέθους 7. Στον πίνακα "*result*" θα αποθηκευτούν τα αποτελέσματα των μετρήσεων κάθε qubit κατά το τέλος του αλγορίθμου.

Στην γραμμή 23 εφαρμόζεται μία πύλη Hadamard σε όλα τα qubit. Για λόγους συντομίας χρησιμοποιείται η μέθοδος «***ApplyToEach()***» η οποία εφαρμόζει μία πύλη σε κάθε στοιχείο ενός register.

Στην γραμμή 25 καλείται η υπορουτίνα "*Oracle*" η οποία θα εκτελέσει το «μαύρο κουτί» με εισόδους την συμβολοσειρά u και την μεταβλητή "*register*".

Στην γραμμή 27 εφαρμόζεται για άλλη μία φορά μία πύλη Hadamard σε όλα τα qubit μέσω της μεθόδου «***ApplyToEach()***».

Στις γραμμές 30-34 πραγματοποιούνται μετρήσεις σε κάθε ένα από τα διαθέσιμα qubit, τα αποτελέσματα των οποίων αποθηκεύονται στην μεταβλητή "*result*".

Στην γραμμή 37 τα qubit που χρησιμοποιήθηκαν επιστρέφουν όλα στην κατάσταση $|0\rangle$ ώστε να είναι δυνατή η επαναχρησιμοποίησή τους.

Τέλος, στην γραμμή 38 επιστρέφονται τα αποτελέσματα των μετρήσεων.

Έχοντας ολοκληρώσει την κατασκευή των υπορουτινών σε Q#, ακολουθεί η δημιουργία του κλασσικού προγράμματος σε C# το οποίο θα καλέσει την υπορουτίνα "*Bernstein_Vazirani*" και θα εξετάσει τα αποτελέσματα. Ακολουθεί ο κώδικας C#.

```
12      static void Main(string[] args)
13      {
14          System.Console.WriteLine();
15          // Run the quantum code on a "co-processor"
16          using (var qsim = new QuantumSimulator())
17          {
18              // Secret string u
19              var u = new long[] { 1, 1, 0, 0, 1, 0, 1 };
20              // Simulate operation Bernstein_Vazirani
21              var received = Bernstein_Vazirani.Run(qsim, new QArray<long>(u)).Result;
22              // Print Results
23              System.Console.Write(" Secret string : ");
24              foreach (var item in u)
25              {
26                  Console.Write(item.ToString());
27              }
28              System.Console.WriteLine();
29              System.Console.Write("      Measured : ");
30              foreach (var item in received)
31              {
32                  Console.Write(item.ToString());
33              }
34              System.Console.WriteLine("\n");
35
36              // Compare arrays "u" and "received"
37              bool equal = u.SequenceEqual(received);
38              if (equal)
39              {
40                  Console.WriteLine(" Secret String was obtained successfully.");
41              }
42          }
43          System.Console.ReadLine();
44      }
```

Εικόνα 5-15. Driver.cs

Στην γραμμή 16 δημιουργείται ένα *instance* του κβαντικού προσομοιωτή του Quantum Development Kit στην μεταβλητή "*qsim*".

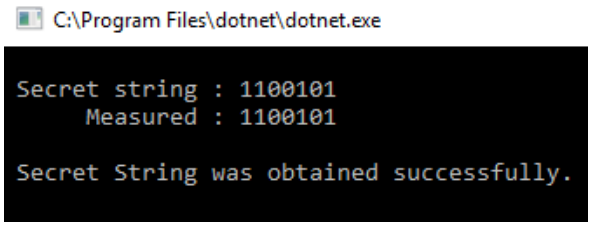
Στην γραμμή 19 αρχικοποιείται ο πίνακας "*u*" ο οποίος αντιστοιχεί στην «άγνωστη» συμβολοσειρά u.

Στην γραμμή 21 η μεταβλητή "*qsim*", χρησιμοποιείται για την προσομοίωση της υπορουτίνας "*Bernstein_Vazirani*" μαζί με τον πίνακα "*u*" ως είσοδο. Ο τύπος μεταβλητής ο οποίος χρησιμοποιείται για τη μετάδοση πινάκων σταθερού μήκους προς και από τον κώδικα Q# είναι ***QArray***. Επομένως, πρέπει να δημιουργηθεί ένα *instance* αυτού του τύπου μεταβλητής από τον εν λόγω πίνακα πριν την μεταβίβασή του («***new QArray<long>(u)***»). Τα αποτελέσματα της υπορουτίνας αποθηκεύονται στην μεταβλητή "*received*".

Στις γραμμές 23-34 τυπώνονται στον χρήστη η κρυφή συμβολοσειρά u και η συμβολοσειρά που προέκυψε από τις μετρήσεις του αλγορίθμου.

Στις γραμμές 37-41 πραγματοποιείται σύγκριση μεταξύ τον δύο συμβολοσειρών και μόνο αν είναι ίδιες τότε τυπώνεται στον χρήστη το μήνυμα "Secret String was obtained successfully.".

Εκτελώντας το πρόγραμμα, προκύπτουν τα εξής αποτελέσματα:



Εικόνα 5-16. Αποτελέσματα αλγορίθμου Bernstein-Vazirani

Παρατηρείται ότι η συμβολοσειρά u ανακτήθηκε επιτυχώς στην ολότητά της.

6 Πίνακας Σύγκρισης

Στο σημείο αυτό, έχοντας ολοκληρώσει μία συνοπτική περιγραφή και για τα δύο κιτ ανάπτυξης κβαντικού λογισμικού, Qiskit και Quantum Development Kit, ακολουθεί ένα πίνακας σύγκρισης μεταξύ των δύο.

	Qiskit	**Quantum Development Kit**	
Εταιρία	IBM	Microsoft	
Κυκλοφορία	19/12/2018	11/12/2017	
Σκοπός	Ανάπτυξη κβαντικού λογισμικού για τις quantum cloud computing υπηρεσίες της IBM Research.	Παροχή ενός πλήρους περιβάλλοντος για ανάπτυξη και προσομοίωση κβαντικών κυκλωμάτων.	
Πλατφόρμες	Jupyter Notebook και IBM Q Experience	Visual Studio 2017 και VSCode	
Tutorials	Qiskit Tutorials	Quantum Kata	
Γλώσσες Προγραμματισμού	**Python**	**Q#** Για την σύνθεση υπορουτινών και την δημιουργία του κβαντικού κυκλώματος.	**C#** Για την επίκληση του κβαντικού προσομοιωτή, την παροχή κλασσικών δεδομένων εισόδου και την ανάγνωση δεδομένων εξόδου από τον προσομοιωτή.
Μοντέλο Προγραμματισμού	1. Ορισμός quantum και classical register στο quantum circuit 2. Σύνθεση κυκλώματος (εφαρμογή κβαντικών πυλών) 3. Επιλογή backend για την εκτέλεση του κυκλώματος 4. Συλλογή αποτελεσμάτων 5. Απεικόνιση αποτελεσμάτων (ιστόγραμμα)	1. Δημιουργία υπορουτίνας **(Q#)** 2. Δέσμευση qubit **(Q#)** 3. Σύνθεση κυκλώματος **(Q#)** 4. Αποδέσμευση qubit **(Q#)** 5. Επιστροφή αποτελεσμάτων **(Q#)** 6. Εκτέλεση υπορουτίνας στον κβαντικό προσομοιωτή **(C#)** 7. Συλλογή αποτελεσμάτων **(C#)**	
Αρχικοποίηση Qubit	Ορισμός QuantumRegister και ClassicalRegister στο QuantumCircuit. «***qr = QuantumRegister(2)***» «***cr = ClassicalRegister(2)***» «***qc = QuantumCircuit(qr, cr)***» Τα qubit αρχικοποιούνται στην κατάσταση $\|0\rangle$.	Δέσμευση των qubit εντός του μπλοκ «*using*». «***using (register = Qubit[2]){…}***» Τα qubit αρχικοποιούνται στην κατάσταση $\|0\rangle$. *Κατά το τέλος των εργασιών απαιτείται η αποδέσμευση των qubit και η επιστροφή τους στην κατάσταση* $\|0\rangle$.	

Οπτικοποίηση Κυκλώματος	Ναι «***qc.draw()***»	Όχι
Εκτέλεση Κβαντικών Κυκλωμάτων	Επιτρέπει την εκτέλεση κβαντικών κυκλωμάτων σε **προσομοιωτές** και σε πραγματικούς **κβαντικούς υπολογιστές**.	Επιτρέπει την εκτέλεση κβαντικών κυκλωμάτων μόνο σε **προσομοιωτές**.
Αριθμός Qubit	Ο προσομοιωτής *ibmq_qasm_simulator* της IBM Q παρέχει έως και 32 qubits. Ο κβαντικός υπολογιστής *ibmq_16_melbourne* της IBM Q παρέχει έως και 14 qubits.	Ο προσομοιωτής του QDK παρέχει έως και 32 qubits. Ο προσομοιωτής της cloud πλατφόρμας Azure παρέχει έως και 40 qubits.
Αναπαράσταση Κβαντικών Καταστάσεων	Το Qiskit αναπαριστά την κβαντική κατάσταση των n qubits με το τελευταίο qubit να βρίσκεται στο αριστερό μέρος και το πρώτο qubit στο δεξί μέρος. $Q_{n-1} \otimes \ldots \otimes Q_1 \otimes Q_0$	-
Μέτρηση	Το αποτέλεσμα μίας μέτρησης ενός qubit (quantum register) καταχωρείται στο classical register. Η ανάκτηση των δεδομένων πραγματοποιείται με ανάγνωση των bits στο classical register. Το αποτέλεσμα μίας μέτρησης επιστρέφει την τιμή *0* ή *1*. «***qc.measure(qr, cr)***»	Το αποτέλεσμα μίας μέτρησης επιστρέφει την τιμή *Zero* ή *One*. «***M (qubit);***»
Αποτελέσματα	Για την απόκτηση των αποτελεσμάτων, επιλέγεται ένα backend το οποίο θα εκτελέσει το κβαντικό κύκλωμα. «***job = execute(qc, backend)***» Για την ανάκτηση των αποτελεσμάτων: «***job.result()***»	Τα αποτελέσματα των μετρήσεων επιστρέφονται ως έξοδος της υπορουτίνας. Δημιουργία ενός instance του κβαντικού προσομοιωτή του QDK (*qsim*) για την προσομοίωση της υπορουτίνας και την ανάκτηση των αποτελεσμάτων. «***subroutine.Run(qsim).Result;***»
Απεικόνιση Αποτελεσμάτων	Ναι π.χ. «***plot_histogram***»	Όχι

7 Βιβλιογραφία

[1] Λ. Περιβολαρόπουλος, Εισαγωγή στην Κβαντική Θεωρία, Εκδόσεις Κάλλιπος, 2015.

[2] J. Al-Khalili, Κβαντικά Παράδοξα, Εκδόσεις Τραυλός, 2005.

[3] M. A. Manucci & Noson S. Yanofsky, Quantum Computing for Computer Scientists, Cambridge University Press, 2008.

[4] M. A. Nielsen & I. L. Chuang, Quantum Computation and Quantum Information, 10th Anniversary Edition, Cambridge University Press, 2010.

[5] M. D. Ciletti & M. M. Mano, Ψηφιακή Σχεδίαση, Τέταρτη έκδοση, Εκδόσεις Παπασωτηρίου, 2007.

[6] Patrick J. Coles, Stephan Eidenbenz, et al. , "Quantum Algorithm Implementations for Beginners," 2018.

[7] Andrew W. Cross, Lev S. Bishop, et al. , "Open Quantum Assembly Language," 2017.

[8] Jiangfeng Du, Mingjun Shi, et al. , "Implementation of a Quantum Algorithm to Solve Bernstein-Vazirani's Parity Problem," 2000.

[9] Α. Χ. Τζέμος, "https://www.freewebs.com/thanasistzemos/," 2008. [Online]. Available: https://www.freewebs.com/thanasistzemos/Tensors%20in%20Quantum%20Information.pdf.

[10] U. Vazirani, "BerkeleyX CS-191x: Quantum Mechanics and Quantum Computation," August 2013. [Online]. Available: https://courses.edx.org/courses/BerkeleyX/CS-191x/2013_August/course/.