Un'introduzione al Quantum Computing

Lectio Magistralis
Olimpiadi di Informatica a Squadre
Bologna, 11/03/2023

Luciano Bononi

Dipartimento di Informatica – Scienza e Ingegneria

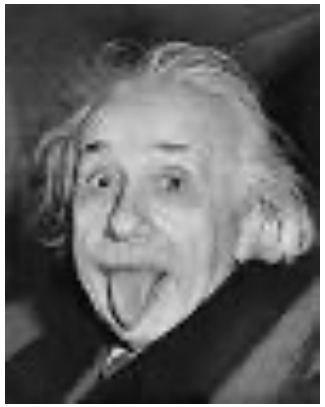
Premessa

Al mondo ci sono solo 10 tipi di persone...

quelle che capiscono l'aritmetica binaria e quelle che non la capiscono.

anonimo informatico

Al termine di questa illustrazione forse ridurremo l'entropia dell'universo informatico o forse no:

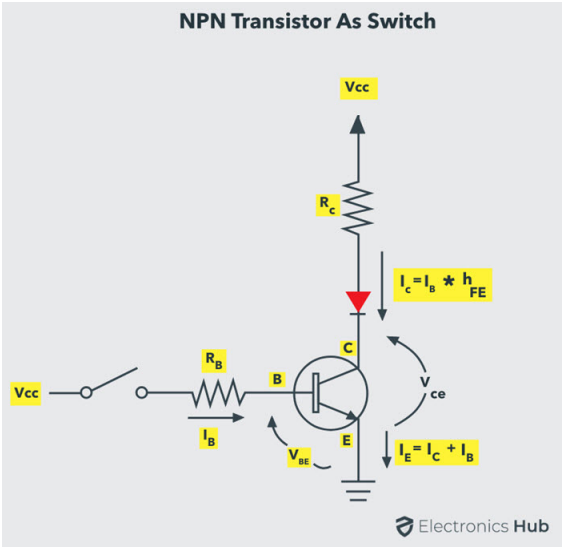


Al mondo ci sono $\binom{0}{1} \otimes \binom{1}{0}$ gruppi di informatici...

suddivisi dal modo in cui capiscono il Quantum Computing.

Disclaimer: le seguenti slide hanno carattere divulgativo e non scientifico in quanto contengono semplificazioni concettuali e intuitive.

dal Computing classico al Quantum Computing (1)



Nel 1947 Walter Brattain, John Bardeen e William Shockley inventarono il transistor (Bell labs 1947, Nobel per la fisica 1956):

APRO o CHIUDO un circuito elettronicamente mediante un segnale di controllo

Corollario: lo stato di un circuito al tempo T è APERTO oppure CHIUSO

0 1

Logica binaria: posso rappresentare l'informazione usando solo due simboli (valori del bit): 0 e 1

Se associo gli Stati dei circuiti Aperto al valore del bit 0 e Chiuso al valore del bit 1 ho una codifica elettronica. Ovvero, lo stato di una macchina che mi rappresenta un valore binario.

Posso costruire macchine che trasformano stati in stati diversi prevedibili? Certo!

Le porte logiche realizzano funzioni di trasformazione di segnali di input in segnali di output secondo una tabella di verità pre-definita e DETERMINISTICA (ovvero, da stesso input e operazione ottengo sempre lo stesso risultato prodotto)

dal Computing classico al Quantum Computing (2)

La porta logica NAND (Not AND) è interessante:

tabella di verità (NAND):

A	B	Q (output)
0	0	1
0	1	1
1	0	1
1	1	0

Si dimostra che componendo NAND posso ottenere tutte le funzioni logiche di base.
Not (x) = 0 se x=1, oppure 1 altrimenti
OR (x,y) = 1 se almeno uno tra x e y è 1, zero altrimenti
AND (x,y) = 1 se sia x che y sono 1, zero altrimenti
XOR (x,y) = 1 se uno solo uno tra x e y è uguale a uno, zero altrimenti
ecc. ecc.

Esistono porte logiche che sono il mattoncino sufficiente a comporre calcoli complessi arbitrari.

dal Computing classico al Quantum Computing (3)

Se creiamo una composizione di porte logiche il funzionamento di un calcolo governabile mediante segnali di controllo binari (il programma eseguibile) e facciamo eseguire il calcolo sui dati di input del nostro problema otteniamo un computer classico programmabile che ci darà i risultati di output attesi per qualsiasi combinazione ammissibile dell'input.

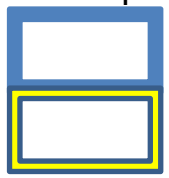
fatto 1: abbiamo un solo stato ad ogni istante, che rappresenta l'evoluzione del processo di calcolo (risultato intermedio) verso la soluzione. Passi sequenziali causali (e deterministici): il risultato del passo successivo può dipendere ANCHE dal risultato intermedio ottenuto dei passi precedenti, ed è univoco e ripetibile.

fatto 2: quando avrò completato il processo di calcolo? Quando il calcolatore avrà eseguito tutti i passi necessari in sequenza. E se per caso sbaglio strada (scrivo un programma errato) per ottenere il calcolo atteso? Otterrei un risultato sbagliato o non sapremo mai il risultato...e non sapremo mai che non lo sapremo mai (problema della terminazione o halting problem è indecidibile).

albero di computazione classica

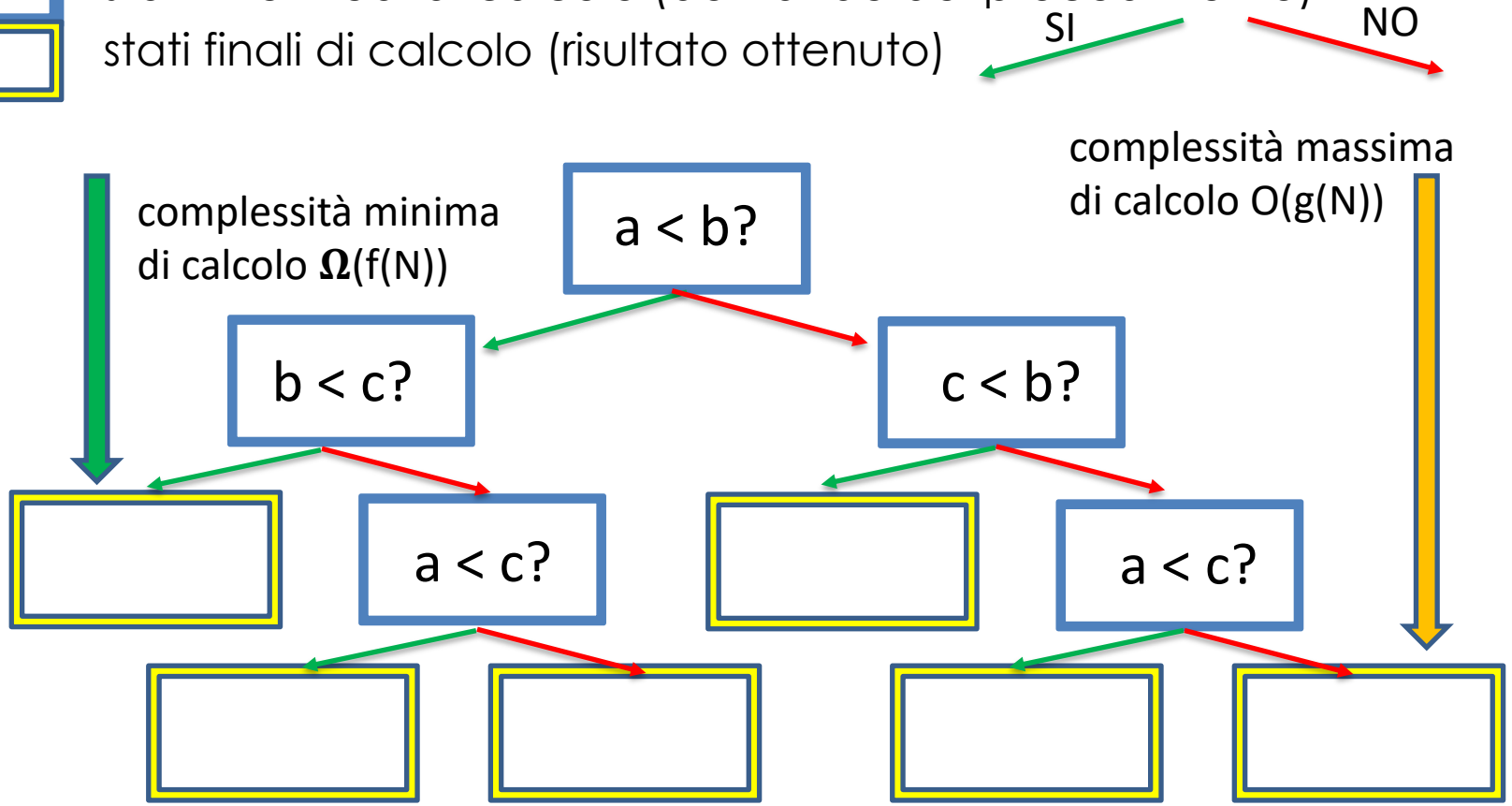
esempio: se voglio ordinare tre elementi a, b, c interi in ordine dal più piccolo al più grande mediante confronti, che programma scrivo?

...e quando lo eseguo cosa succede? e se devo ordinare N elementi?



stati intermedi di calcolo (domande del procedimento)

stati finali di calcolo (risultato ottenuto)



Devo eseguire l'intero percorso computazionale dalla radice dell'albero fino a una foglia (risultato) scegliendo che strada seguire ad ogni scelta, e sperando di non sbagliare per non dovere risalire e ridiscendere da altre strade.

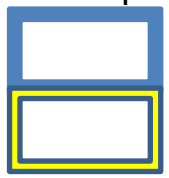
Se devo salire e scendere nell'albero (il programma è poco efficiente o sbagliato) il cammino di calcolo si allunga e potrebbe non arrivare mai al risultato.

<u>Devo fare programmi corretti e efficienti per minimizzare il cammino computazionale</u>

albero di computazione classica

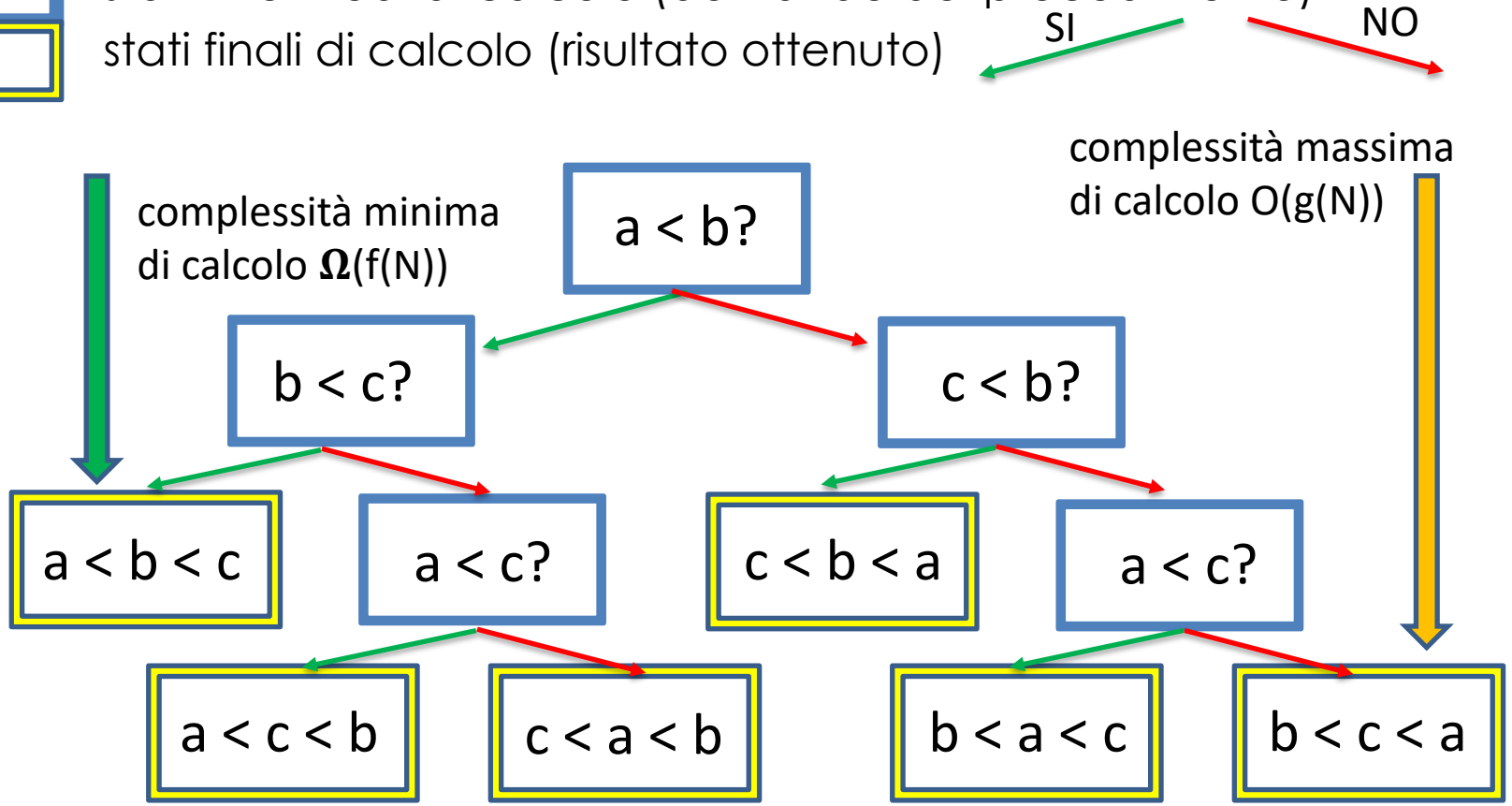
esempio: se voglio ordinare tre elementi a, b, c interi in ordine dal più piccolo al più grande mediante confronti, che programma scrivo?

...e quando lo eseguo cosa succede? e se devo ordinare N elementi?



stati intermedi di calcolo (domande del procedimento)

stati finali di calcolo (risultato ottenuto)



Devo eseguire l'intero percorso computazionale dalla radice dell'albero fino a una foglia (risultato) scegliendo che strada seguire ad ogni scelta, e sperando di non sbagliare per non dovere risalire e ridiscendere da altre strade.

Se devo salire e scendere nell'albero (il programma è poco efficiente o sbagliato) il cammino di calcolo si allunga e potrebbe non arrivare mai al risultato.

<u>Devo fare programmi corretti e efficienti per minimizzare il cammino computazionale</u>

dal Computing classico al Quantum Computing (4)

Quindi oggi abbiamo computer classici e cerchiamo di programmarli bene, ma dobbiamo attendere tanti passi prima di avere (forse) i risultati.

Ideona! Facciamo passi veloci! ovvero usiamo Clock veloci.

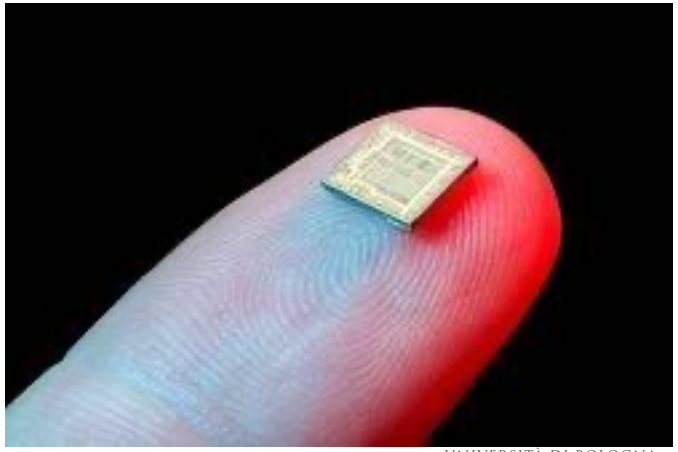
Serve un batterista della «band di calcolo» del chip ancora più veloce.

Ma per farlo abbiamo bisogno di porte logiche più veloci.

Limite fisico: velocità della luce (circa c = 300.000.000 m/sec).

Supponiamo che un calcolo richieda una sequenza di un miliardo di porte logiche da percorrere su un Chip per trasformare i segnali in sequenza da input a risultati di output. Ogni porta logica occupa 10 nanometri. Quanta strada percorre la corrente elettrica passo dopo passo ?

10 nanometri di strada per porta x 1 miliardo di porte = 10 metri



Anche viaggiando a velocità c non si potrà impiegare meno di :

tempo = spazio / velocità = 10 / 300.000.000 = 33,3 nanosecondi.

Ma noi abbiamo ancora più fretta, quindi?

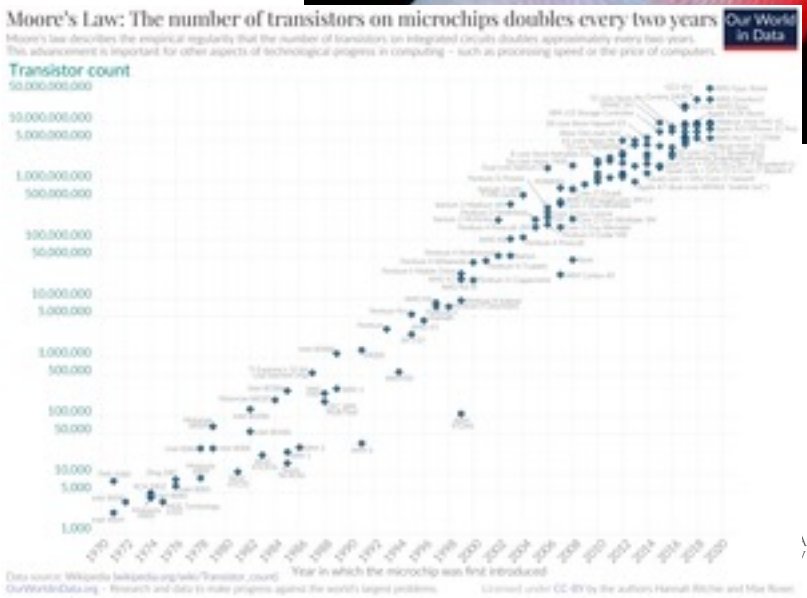
dal Computing classico al Quantum Computing (5)

Ma se non possiamo mandare segnali più veloci della luce, l'unico modo di fare passi più veloci lungo il percorso di calcolo è ridurre la strada totale da percorrere.

Ideona! Miniaturizziamo di più le porte logiche sul Chip.

Legge di Moore: numero di transistor nello stesso spazio su chip raddoppia ogni due anni (nel 2024 saremo intorno a 3-2 nanometri)

Ma esiste un limite fisico che raggiungeremo prima o poi.
Potremo mai fare transistor su meno di un atomo di chip?



Quindi?
Usiamo l'inventiva... parallelizziamo i processi di calcolo dove possibile, ecc. (non è possibile/facile per molti problemi)

limiti fisici della computazione classica

Non potremo accelerare i clock oltre un limite legato alla dimensione fisica dei chip

Non potremo miniaturizzare i chip all'infinito

Non potremo sempre inventare algoritmi e soluzioni parallele più veloci

Ci siamo arenati?.... ma esiste un nuovo modo.

Quantum Computing

Disclaimer: le seguenti slide hanno carattere divulgativo e non scientifico in quanto contengono semplificazioni concettuali e intuitive.

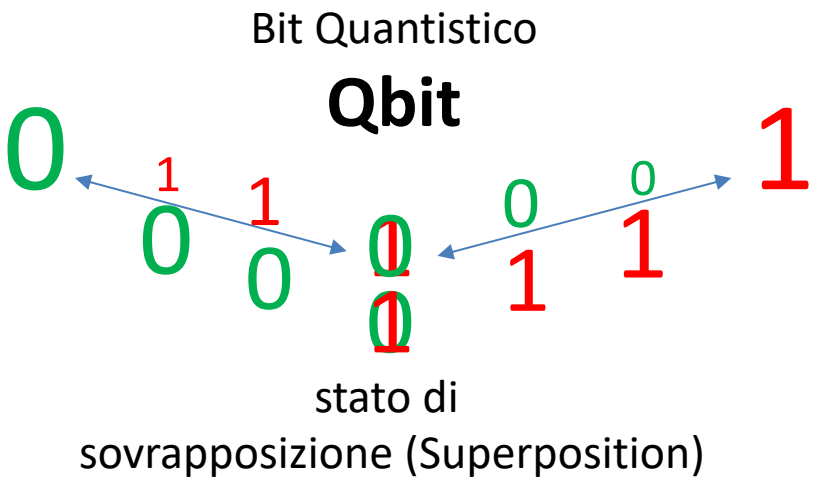
dal Computing classico al Quantum Computing (5)

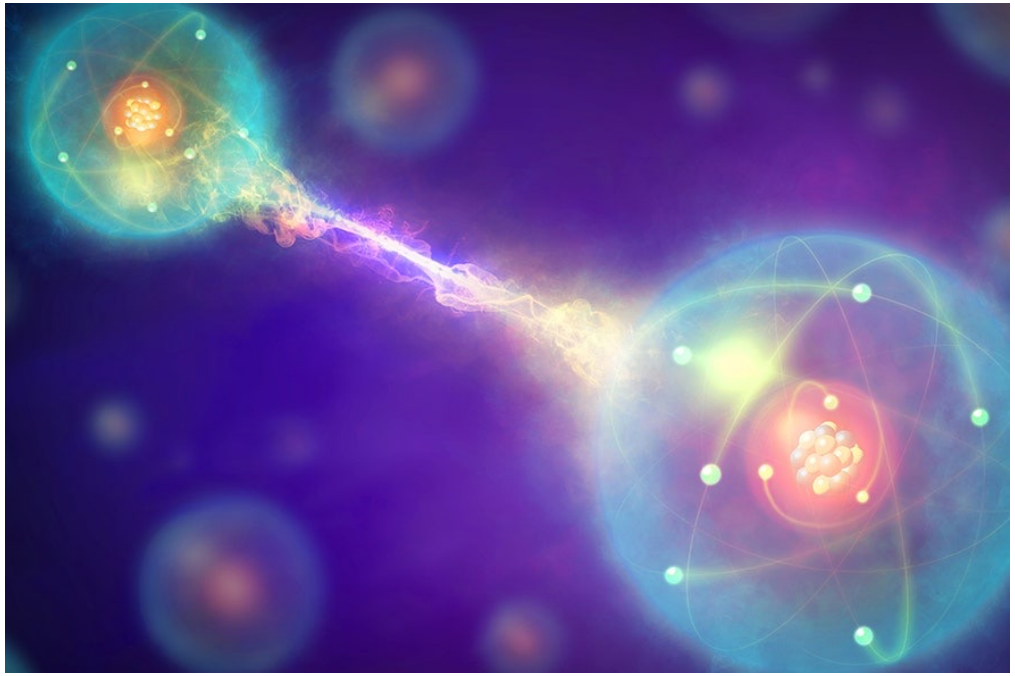
Sfruttiamo allora una nuova FISICA: meccanica quantistica.

- ordine dimensionale delle particelle (sub-atomiche) costituenti l'atomo (da 10^{-15} in giù). Es. elettroni, fotoni, ecc.

- Un universo fatto di fenomeni fisici sorprendenti.

- nuovi bit (Qbit), che in realtà hanno i due stati 0 e 1, ma possono essere in entrambi gli stati contemporaneamente (principio di sovrapposizione).





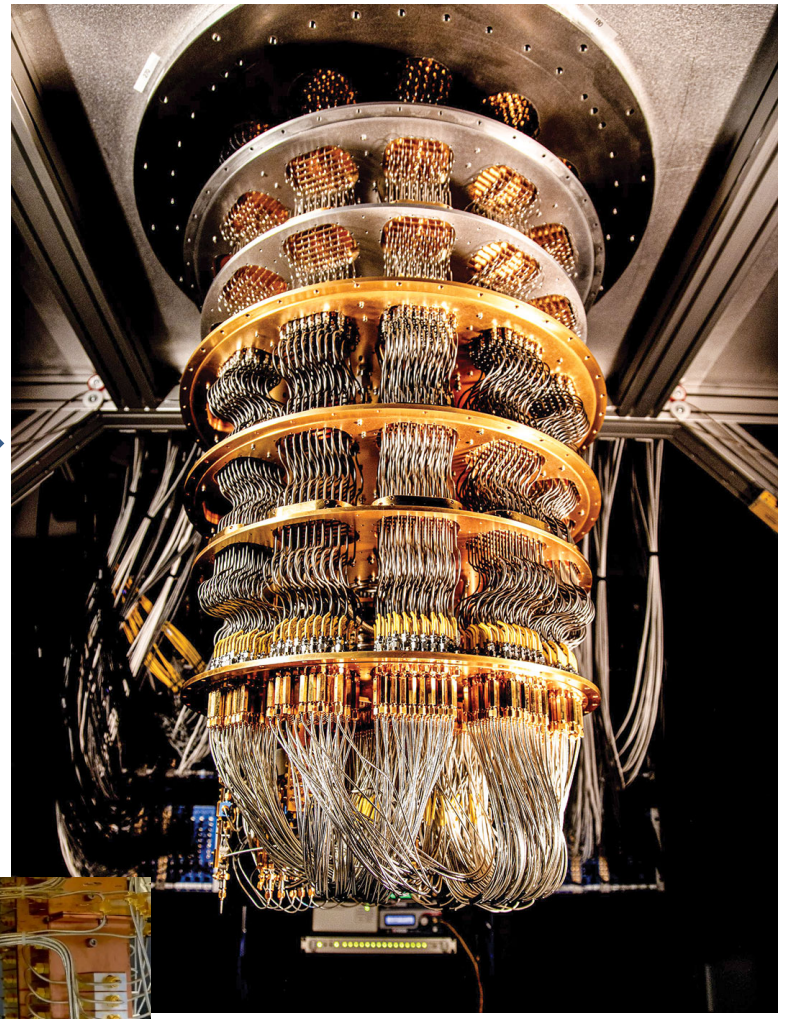
credits: wired italia

dal Computing classico al Quantum Computing (5)

Ma riusciamo a controllare i Qbit in modo analogo alle correnti con un transistor?

- Si, ma dobbiamo agire su superfluidi di particelle vicine allo Zero assoluto (che siano stabili e senza «rumore»).

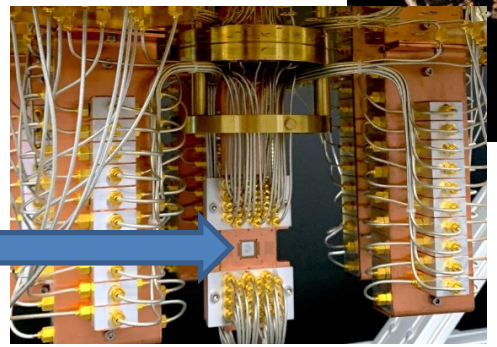
Quantum Computer: prima di tutto un grande FRIGORIFERO.



Riusciamo quindi a costruire nuove funzioni o «porte logiche» del Quantum Computing dalla cui composizione sia possibile realizzare la programmazione di algoritmi quantistici?

- Si, ma sfruttando un nuovo approccio di programmazione probabilistico, legato a fenomeni quantistici «inspiegabili» ma dimostrabili quali tunnel quantistico, Entanglement, Sovrapposizione, ecc.

Un nuovo processore quantistico (dentro al frigorifero)

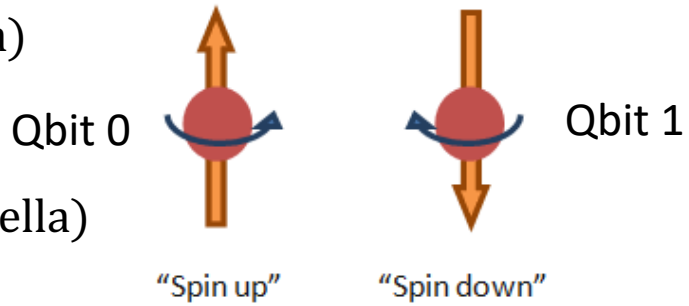


Quantum Computing (approccio algebrico di descrizione dei Qbit)

1) come rappresentare dei vettori di bit del computer classico come Qbits? $|Dirac\ Vector\ Notation\rangle$

vettore che emula Qbit 0 $\stackrel{\text{def}}{=} \begin{pmatrix}1\\0\end{pmatrix} = |0\rangle$ (stato di Spin Up della particella)

vettore che emula Qbit 1 $\stackrel{\text{def}}{=} \begin{pmatrix}0\\1\end{pmatrix} = |1\rangle$ (stato di Spin Down della particella)



Ripasso di regole algebriche di moltiplicazione tra matrici e vettori

$$\begin{pmatrix}a & b\\ c & d\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix} = \begin{pmatrix}ax+by\\ cx+dy\end{pmatrix}$$

$$\begin{pmatrix}a & b & c\\ d & e & f\\ g & h & i\end{pmatrix}\begin{pmatrix}x\\ y\\ z\end{pmatrix} = \begin{pmatrix}ax+by+cz\\ dx+ey+fz\\ gx+hy+iz\end{pmatrix}$$

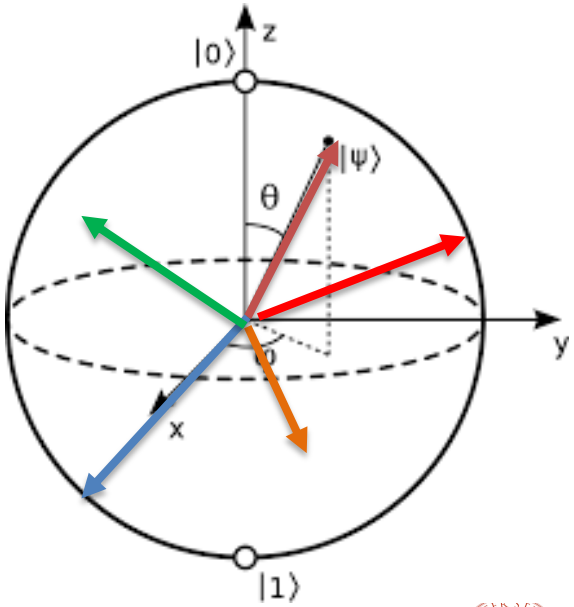
$$\begin{pmatrix}a & b\\ c & d\end{pmatrix}\begin{pmatrix}w & x\\ y & z\end{pmatrix} = \begin{pmatrix}aw+by & ax+bz\\ cw+dy & cx+dz\end{pmatrix}$$

Matrice Identità I

$$\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}a\\b\\c\\d\end{pmatrix} = \begin{pmatrix}a\\b\\c\\d\end{pmatrix}$$

$$\begin{pmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}0\\1\\0\\0\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}$$

inversione righe centrali di I



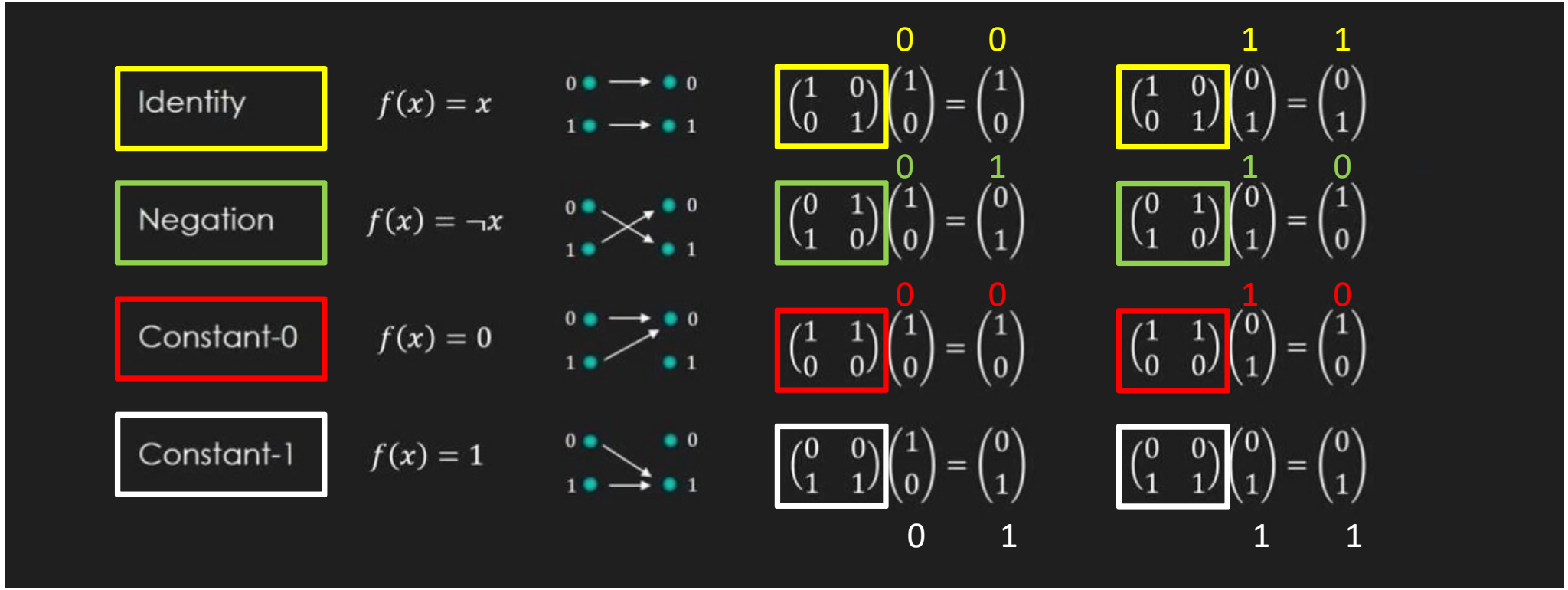
Sfera di Bloch
(possibili infiniti stati di Spin)

Quantum Computing (operatori su un singolo Cbit)

2) quali sono le quattro possibili operazioni che possiamo fare su un SINGOLO bit?
ricordiamo che

vettore Qbit 0 $\stackrel{\text{def}}{=} \begin{pmatrix}1\\0\end{pmatrix} = |0\rangle$

vettore Qbit 1 $\stackrel{\text{def}}{=} \begin{pmatrix}0\\1\end{pmatrix} = |1\rangle$



credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (reversibilità della computazione)

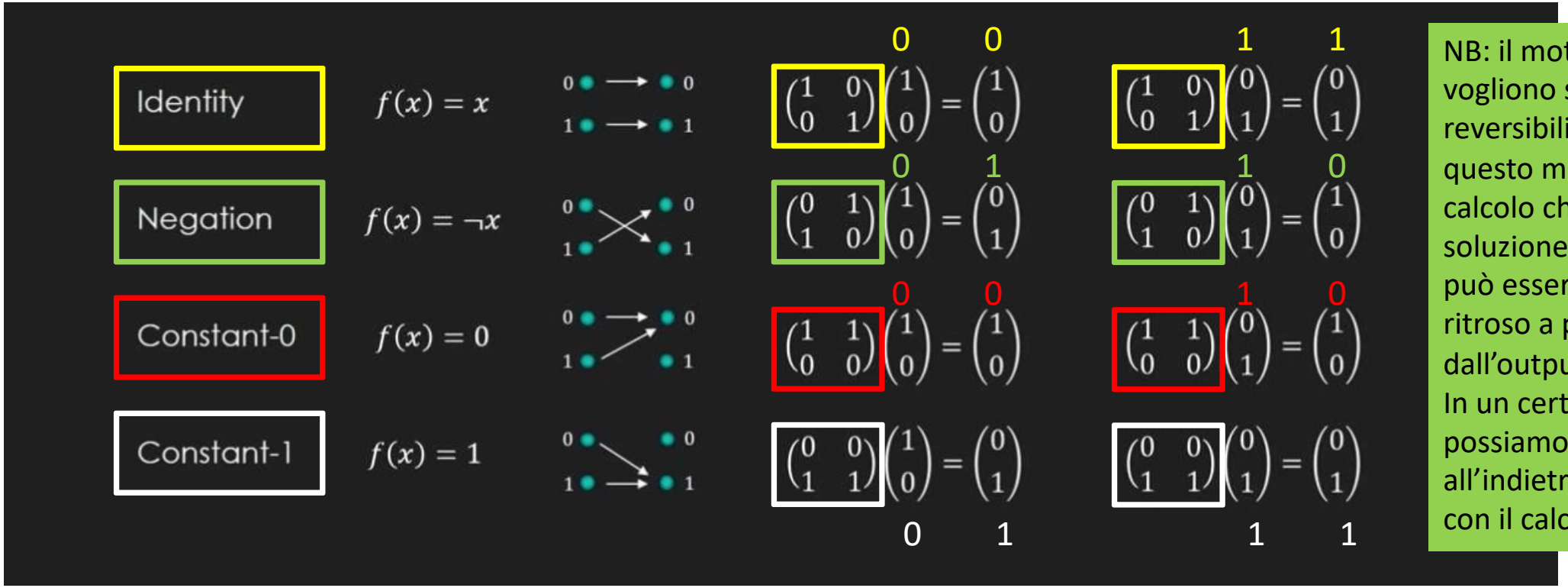
3) quali operatori sono reversibili? dato output e operatore è possibile risalire al valore del Qbit di input

reversibili: identità e negazione (mantengono o permutano il valore di input sull'output)

non reversibili: Const-0 e Const-1 (cancellano il valore di input e lo sovrascrivono sempre con 0 o 1)

NB: qui nella figura stiamo usando i vettori che emulano algebricamente un Qbit

NB: Computer quantistici implementano solo il calcolo con operatori su Qbit che siano reversibili



NB: il motivo per cui si vogliono solo operatori reversibili è perchè in questo modo tutto il calcolo che produce la soluzione da un input può essere percorso a ritroso a partire dall'output.
In un certo senso possiamo «andare all'indietro nel tempo» con il calcolo....

credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (prodotto tensoriale di vettori)

4) il tensor product di due vettori ha le proprietà che vediamo qui sotto.

Ma notiamo che se facciamo prodotto tensoriale di N vettori (che emulano N Qbit) otteniamo una rappresentazione di stato del corrispondenti valori binari possibili espresso dai relativi N Qbit associati.

Ma a che serve il prodotto tensoriale? a rappresentare sequenze di più Qbit (emulati) consecutivi.

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \otimes \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_0 \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \\ x_1 \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} x_0 y_0 \\ x_0 y_1 \\ x_1 y_0 \\ x_1 y_1 \end{pmatrix}$$

NB: il tensor product produce stati di enumerazione binaria di due bit

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \otimes \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 6 \\ 8 \end{pmatrix}$$

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \otimes \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \otimes \begin{pmatrix} z_0 \\ z_1 \end{pmatrix} = \begin{pmatrix} x_0 y_0 z_0 \\ x_0 y_0 z_1 \\ x_0 y_1 z_0 \\ x_0 y_1 z_1 \\ x_1 y_0 z_0 \\ x_1 y_0 z_1 \\ x_1 y_1 z_0 \\ x_1 y_1 z_1 \end{pmatrix}$$

NB: (x1,y1,z1) è un valore binario su tre bit

NB: il tensor product produce stati di enumerazione binaria di tre bit

$$\begin{matrix} 1 & 1 & 0 \end{matrix}$$
$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \begin{matrix} \mathbf{0} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{5} \\ \mathbf{6} \\ \mathbf{7} \end{matrix}$$

credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (prodotto tensoriale di Qbit multipli in sequenza)

5) il tensor product di più vettori (Qbit) serve a rappresentare il product state di più Qbit consecutivi.

vediamo l'esempio di tutte le 4 combinazioni possibili di prodotto tensoriale (product state)
di 2 Qbit $|00\rangle$ $|01\rangle$ $|10\rangle$ $|11\rangle$ oppure di 3 Qbit $|000\rangle$ $|001\rangle$ $|010\rangle$ $|011\rangle$ $|100\rangle$ $|101\rangle$ $|110\rangle$ $|111\rangle$

Stiamo cioè producendo tutti i risultati possibili di stato di due Qbit in cascata (combinazione binaria).

$$|00\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \qquad |01\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$$

$$|4\rangle = |100\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\0\\0\\1\\0\\0\\0\end{pmatrix}$$

$$|10\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \qquad |11\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

- We call this tensored representation the **product state**
- We can **factor** the product state back into the **individual state** representation
- The product state of n bits is a vector of size 2^n

credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (prodotto tensoriale di Qbit multipli in sequenza)

Immaginiamo ora che il vettore di stato che elenca tutti i possibili stati (combinazioni di valori binari) degli N Qbit contenga in ogni riga la probabilità che la combinazione binaria dei valori degli N Qbit sia proprio quella identificata dalla riga. Qui ad esempio, la probabilità di avere risultato binario 4 è 100%, mentre tutti gli altri stati hanno 0%.

$$|00\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \qquad |01\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$$

$$|4\rangle = |100\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\0\\0\\1\\0\\0\\0\end{pmatrix}$$

$$|10\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \qquad |11\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

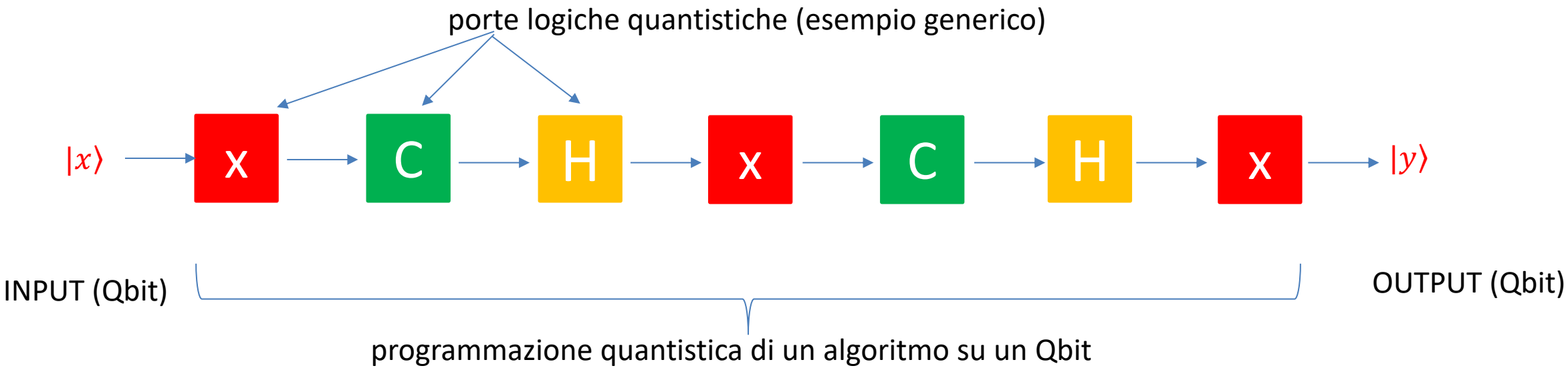
- We call this tensored representation the **product state**
- We can **factor** the product state back into the **individual state** representation
- The product state of n bits is a vector of size 2^n

credits: Quantum Computing for Computer Scientists, Microsoft

Come manipolare i valori dei Qbit? porte logiche quantistiche....

A questo punto, siamo in grado di costruire porte logiche quantistiche che possano manipolare le caratteristiche fisiche delle particelle associate allo stato di Qbit?

Se fossimo in grado di farlo, avremmo delle operazioni di «trasformazione» di input in output sui valori dei Qbit, e potremmo usarle per scrivere dei programmi quantistici.



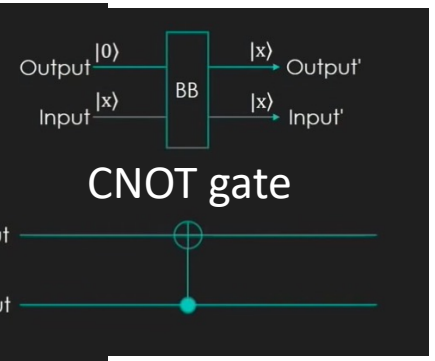
Quantum Computing (la prima porta logica quantistica CNOT)

C

6) finalmente abbiamo un modo per creare la prima porta logica del Computer Quantistico (il mattone iniziale simile al NAND del computer classico e simile a scelta condizionale): **la porta CNOT.**

La porta CNOT esegue un calcolo quantistico (operazione) su 2 Qbit (c, t) che possiamo pensare come segue: dato INPUT (c,t), se c=0 allora Output rimane uguale a (c,t), altrimenti solo t viene invertito e Output = (c, t negato).

- Operates on pairs of bits, one of which is the "control" bit and the other the "target" bit
- If the control bit is 1, then the target bit is flipped
- If the control bit is 0, then the target bit is unchanged
- The control bit is always unchanged
- With most-significant bit as control and least-significant bit as target, action is as follows:



INPUT
(Control, Target)

OUTPUT
(Control, Result)

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

NB: questa matrice implementa esattamente il comportamento di CNOT: se applicata al prodotto tensoriale dei due Qbit di input, produce i due Qbit di output attesi.

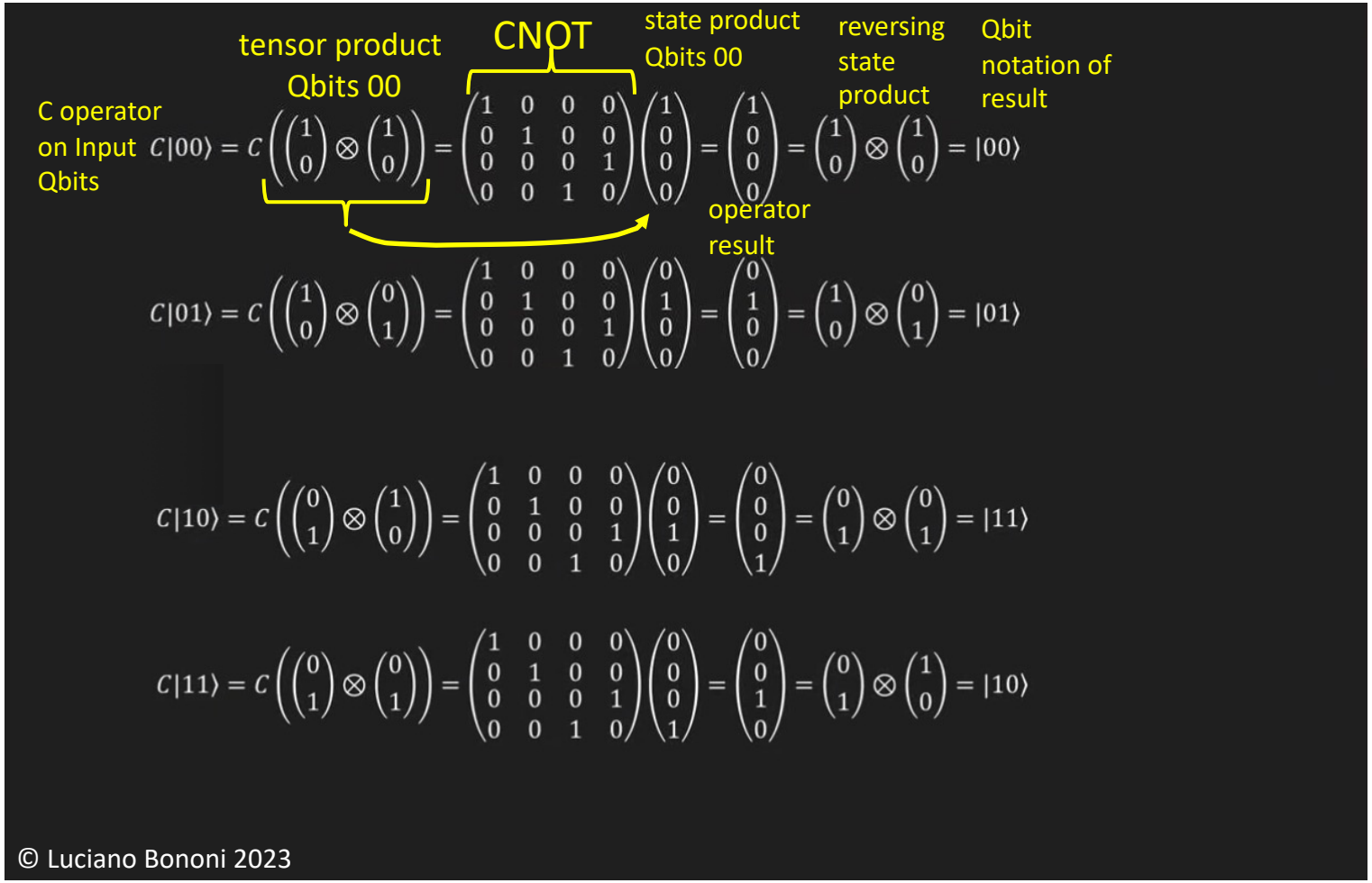
credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (la prima porta logica quantistica CNOT)

C

7) vediamo un paio di casi applicati di cosa succede applicando CNOT (C) a due coppie di Qbit:

NB: il product state prevede tutte le combinazioni possibili del risultato, ma solo una è vera, e facendo il reversing dello state product otteniamo il risultato in Qbit.



Applichiamo CNOT a $|00\rangle$ (control = 0 e target = 0) quindi deve produrre $|00\rangle$

Applichiamo CNOT a $|01\rangle$ (control = 0 e target = 1) quindi deve produrre $|01\rangle$

Applichiamo CNOT a $|10\rangle$ (control = 1 e target = 0) quindi deve produrre $|11\rangle$

Applichiamo CNOT a $|11\rangle$ (control = 1 e target = 1) quindi deve produrre $|10\rangle$

credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (sovrapposizione quantistica)

8. Vediamo da ora in poi che in effetti stiamo usando Qbits (perchè i vettori usati fino ad ora sono in effetti emulazioni discrete pari ai valori estremi 0 e 1 dei Qbit reali).

I Qbit reali infatti sono rappresentabili come $\begin{pmatrix} a \\ b \end{pmatrix}$ con a,b coefficienti complessi t.c. $\|a\|^2 + \|b\|^2 = 1$

I vettori usati fino ad ora $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ e $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ rispecchiano infatti la definizione di Qbit.

Altri esempi di Qbit interessanti: (è verificabile che $\|a\|^2 + \|b\|^2 = 1$)

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \qquad \begin{pmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{pmatrix} \qquad \begin{pmatrix} -1 \\ 0 \end{pmatrix} \qquad \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}$$

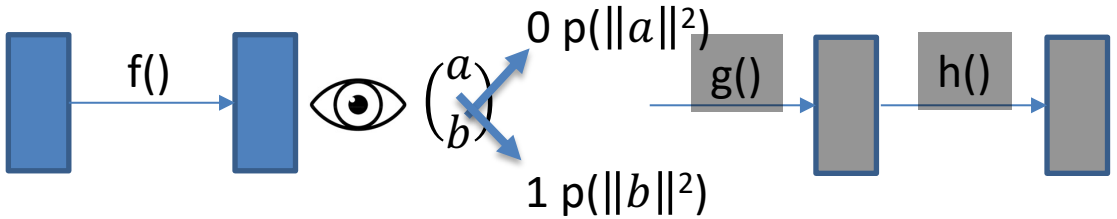
Notare che possiamo usare coefficienti irrazionali e negativi (e anche complessi, ma non qui).

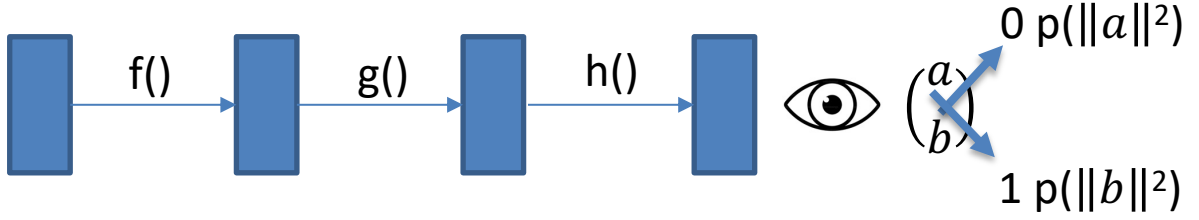
Quantum Computing (sovrapposizione quantistica)

8. Vediamo da ora in poi che possiamo mettere due o più Qbits in sovrapposizione (superposition).

In tale condizione il valore del Qbit non è più necessariamente solo 0 o 1, ma è un po' Zero e un po' Uno allo stesso tempo.

Fatto: non sappiamo quanto un Qbit sia zero o uno durante la computazione quantistica fino a che non MISURIAMO il Qbit, ma quando lo misuriamo COLLASSIAMO il suo valore reale su uno dei due valori ideali Zero oppure Uno con una certa probabilità, e da questo momento il Qbit sarà consumato per sempre. Per questo motivo, a volte misuriamo i Qbit solo al termine del processo di calcolo (non durante), perchè misurandoli collassano e si CONSUMANO.





La scatola blu contiene il Qbit, che viene elaborato dall'algoritmo, ma non sappiamo il valore.
Solo quando osserviamo il valore (apriamo la scatola) troviamo il valore, ma distruggiamo il Qbit.
Ricorda il paradosso del Gatto di Schrödinger.

Misura del Qbit (osservazione): Se un Qbit viene misurato mentre è nello stato $\begin{pmatrix} a \\ b \end{pmatrix}$ allora esso collasserà nel valore risultato 0 con probabilità $\|a\|^2$ e nel valore risultato 1 con probabilità $\|b\|^2$

Quantum Computing (sovrapposizione quantistica)

Ad esempio, il Qbit nello stato $\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$ avrà $\left\|\frac{1}{\sqrt{2}}\right\|^2 = 1/2$ probabilità di collassare nel valore 0, oppure 1.

E quindi il Qbit $\binom{0}{1}$ e $\binom{1}{0}$ quando vengono misurati che valore di computazione producono e con quale probabilità?

$\binom{0}{1}$ collassa su 1 con probabilità $\|b\|^2 = 1^2 = 100\%$ (infatti per noi è stato usato come Qbit $|1\rangle$)

$\binom{1}{0}$ collassa su 0 con probabilità $\|a\|^2 = 1^2 = 100\%$ (infatti per noi è stato usato come Qbit $|0\rangle$)

Quindi, in sostanza, non possiamo più fare controlli intermedi sui valori calcolati? FALSO

Se facciamo controlli consumiamo dei Qbit che non potremo più usare avendoli fatti collassare.

Però in realtà non dobbiamo più scegliere una sola strada dell'albero di scelte, ma scendiamo tutte le strade contemporaneamente (quindi servono meno controlli)!

Grande Vantaggio Computazionale (se il problema si presta) perchè avremo ottenuto nei Qbit tutte le soluzioni possibili del problema (e non solo una) in un numero massimo di passi di calcolo congruo all'altezza dell'albero di computazione (quindi di tipo logaritmico rispetto al numero di stati della computazione).

Quantum Computing (sovrapposizione quantistica)

Allo stesso modo, Qbit multipli sono risolti in modo simile come prodotto di tensori (⊗).

Ad esempio: se volessimo tirare due monete insieme (testa o croce) e volessimo calcolare tutti i possibili risultati, basta programmare due Qbits che producano testa o croce al 50% e farne il prodotto dei tensori:

$$\underset{A}{\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}} \otimes \underset{B}{\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}} = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

(notare che $\|1/2\|^2 = ¼$, e che $¼ + ¼ + ¼ + ¼ = 1$.

Quindi misurando i due Qbit A e B dopo l'esecuzione dei lanci si vedranno collassati entrambi in uno state vector che mostra uno solo dei risultati $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ con ¼ di probabilità ognuno.

Tale calcolo è coerente con tutti i risultati che potremmo ottenere da infiniti lanci di due monete reali.

Quantum Computing (manipolazione di Qbits)

A questo punto possiamo manipolare il calcolo (implementando il programma o algoritmo quantistico) sui Qbits (struttura dati dell'algoritmo) allo stesso modo visto per i vettori di emulazione, ovvero mediante operatori (sotto forma di matrice).

Abbiamo operatori es. CNOT, ecc. e molti di questi hanno senso solo nel contesto quantistico.

Gli operatori in forma di matrice modellano gli effetti di dispositivi fisici che nella realtà quantistica manipolano lo Spin/Polarizzazione delle particelle del super-fluido, ma senza misurarne/osservarne il valore e quindi senza collassarli e consumarli.

Ad esempio: questo è l'operatore programmato per fare il bit-flip di un Qbit: $X \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} b \\ a \end{pmatrix}$

$= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

$$\text{esempio}: \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{3}}{2} \\ \frac{1}{2} \end{pmatrix}$$

(questo Qbit (rosso) era Uno con 3/4 e Zero con ¼ di probabilità, ma dopo il bit flip è un Qbit modificato (blu) con valori Uno con 1/4 e Zero con 3/4 di probabilità.

Intuitivamente, se a un certo punto del calcolo il Qbit avesse un valore zero o uno con certe probabilità, senza osservarlo o consumarlo, il bit flip 

trasforma il Qbit (cambia il suo stato) assumendo le stesse probabilità di assumere i valori uno e zero, ma in modo invertito.

Quantum Computing (Hadamard Gate)

H

Un operatore fondamentale per la programmazione del calcolo quantistico è l'Hadamard Gate.

Tale operatore H prende in input un Qbit (con valore qualsiasi, anche 0 o 1) e lo pone nello stato di esatta sovrapposizione (ovvero in equilibrio statistico tra i due stati 0 e 1, con probabilità ½ di collassare sul valore 0 e ½ di collassare sul valore 1). Equivale a una specie di reset dello stato di un Qbit che potrà avanzare nel calcolo essendo un po' 0 e un po' 1, ma la cosa importante è che in realtà il Qbit sa come tornare indietro (l'operazione è reversibile).

Hadamard gate H = $\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}$ (Notare il valore negativo. Perchè secondo voi serve quel segno?

Perchè rende H reversibile! Fare la prova per vedere H H $|0\rangle = |0\rangle$ e H H $|1\rangle = |1\rangle$. Notevole!

- The Hadamard gate takes a 0- or 1-bit and puts it into exactly equal superposition

$$H|0\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \qquad H|1\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}$$

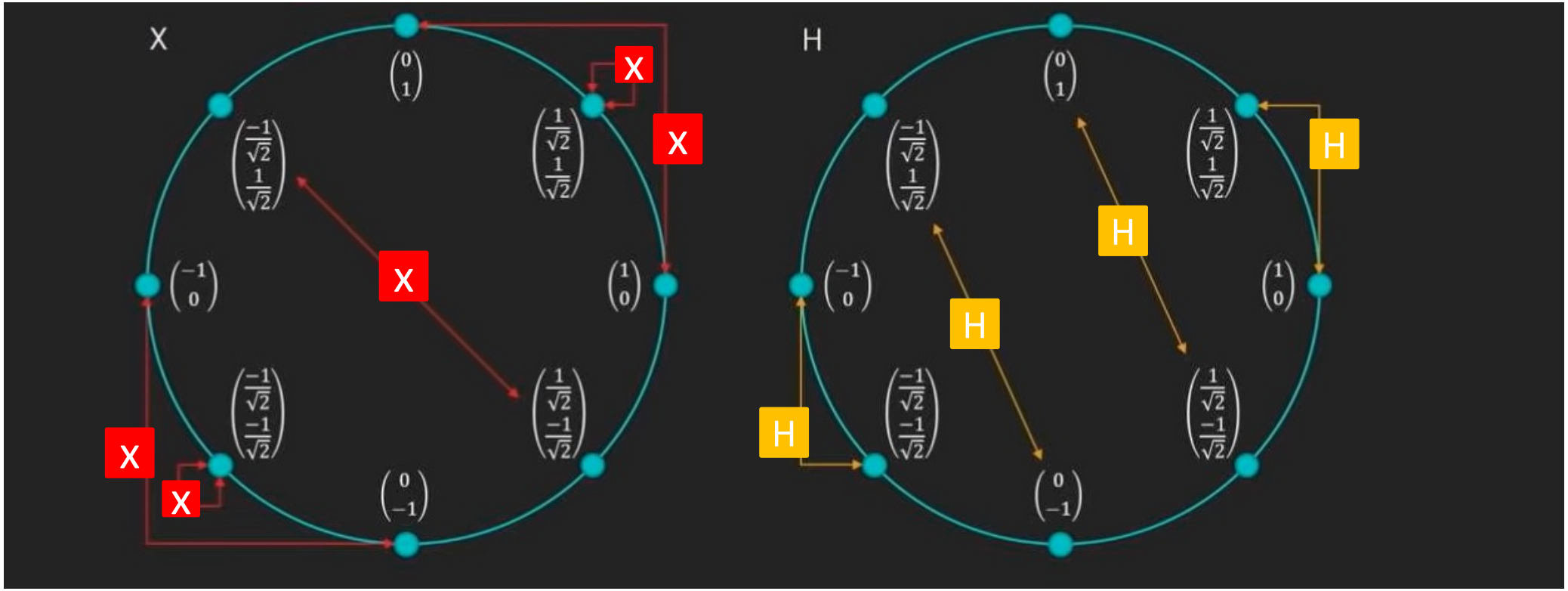
credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (Unit Circle State Machine)

Possiamo ora riassumere gli stati interessanti e le transizioni che siamo in grado di realizzare per la programmazione di un Qbit. La mappa dei suoi possibili stati di Spin rappresentati sullo Unit Circle (una proiezione 2D della sfera di Bloch) per noi informatici equivale a guardare gli stati e le transizioni di una macchina a stati programmabile.

Bit Flip Operator (X)

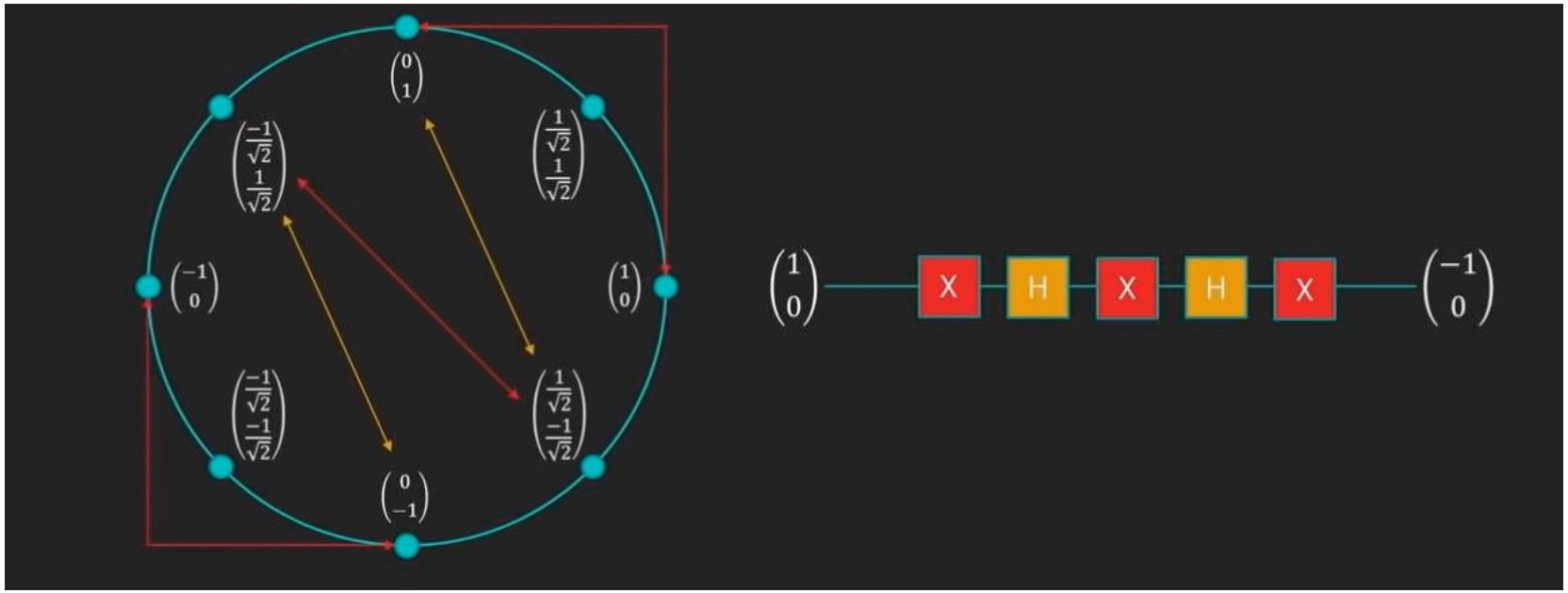
Hademard Operator (H)



credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (Quantum Circuit Notation)

Come possiamo immaginare un calcolo quantistico in un algoritmo implementato mediante operatori in sequenza a partire da un Qbit iniziale di input e produrre un risultato di Qbit risultato?
Attraverso una navigazione della macchina a stati che segua i passi di trasformazione del Qbit.
Notare che il processo di calcolo è invertibile (il calcolo avviene nei due sensi).
Da $\binom{1}{0}$ facciamo BitFlip (X) e arriviamo a $\binom{0}{1}$ e qui facciamo Hademard H che resetta in sovrapposizione allo stato $\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}$ poi di nuovo X che conduce a $\begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$, poi H che porta a $\binom{0}{-1}$ e infine X che porta a $\binom{-1}{0}$



credits: Quantum Computing for Computer Scientists, Microsoft

Quantum Computing (Riassunto)

Riassumendo tutto quello che abbiamo visto:

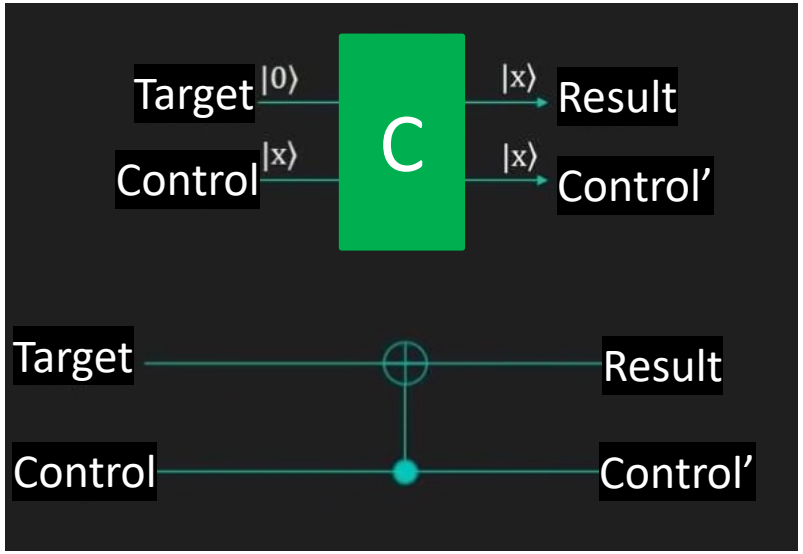
- abbiamo modellato i Qbits come vettori binari di valori complessi
- i bit classici che assumono solo i valori canonici 0 e 1 li chiamiamo Cbits.
- i Cbits sono casi particolari di Qbits e possono essere modellati come vettori di 2 numeri complessi
- Qbits possono essere in sovrapposizione, e se misurati collassano probabilisticamente in Cbits, con certe probabilità che dipendono dallo stato quantistico raggiunto del Qbit.
- Sistemi composti da Qbit multipli sono prodotti tensoriali di sistemi a singolo Qbit (come Cbits in sequenza)
- Gli operatori sui Qbits (espressi come vettori) sono modellabili come matrici moltiplicative (come sui Cbits)
- l'Hadamard gate porta Qbits in equa sovrapposizione, e li riporta indietro al valore originale
- i Qbits e le loro operazioni possono essere interpretati come algoritmi in esecuzione su macchina a stati (su unit circle, se usiamo valori reali, e sulla sfera di Bloch se usiamo anche valori complessi).

«Algoritmo» (CNOT) C

CNOT gate: Control	Target	Control'	Result
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

NB: qui in (Control == 1) noi osserviamo il Qbit Control !
Ma prima lo avevamo copiato per propagarlo in avanti

```
Control' = Identity (Control)
If (Control == 1) then
            result = Negation (Target)
              else
            result = Identity(Target)
```



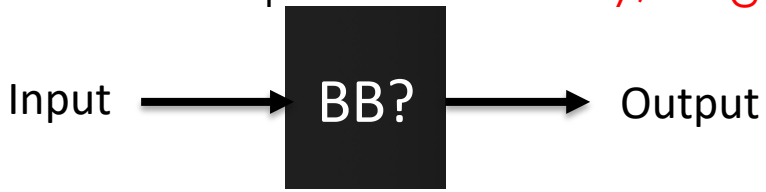
credits: Quantum Computing for Computer Scientists, Microsoft

Questa slide serve solo a fare capire che la porta quantistica CNOT è in realtà utilizzabile secondo un principio di programmazione classica che ci è più «riconoscibile».

The Deutsch Oracle problem (o problema della funzione «misteriosa»)

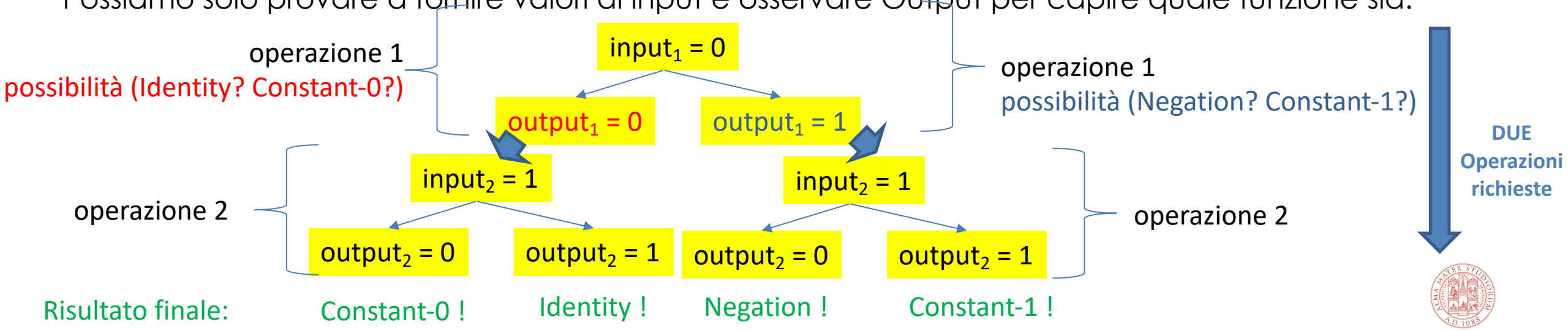
Immaginiamo che ci venga fornita una scatola nera (Black Box = BB) che implementa una funzione su un bit.

Sappiamo che la funzione sarà una delle 4 possibili: Identity, Negation, Constant-0, Constant-1, ma di quale si tratta?



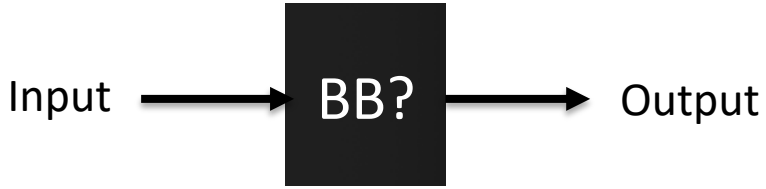
Problema: quante operazioni è necessario fare per sapere la risposta su un **Computer Classico?**

Possiamo solo provare a fornire valori di input e osservare Output per capire quale funzione sia.



The Deutsch Oracle problem

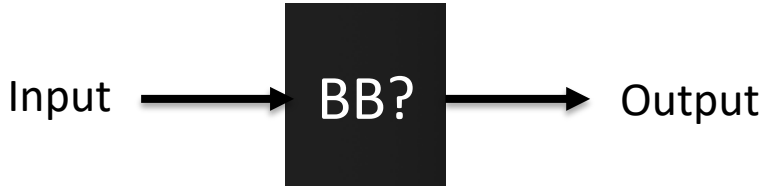
Immaginiamo che ci venga fornita una scatola nera che implementa una funzione su un bit. Sappiamo che la funzione sarà una delle 4 possibili: Identity, Negation, Constant-0, Constant-1, ma di quale si tratta?



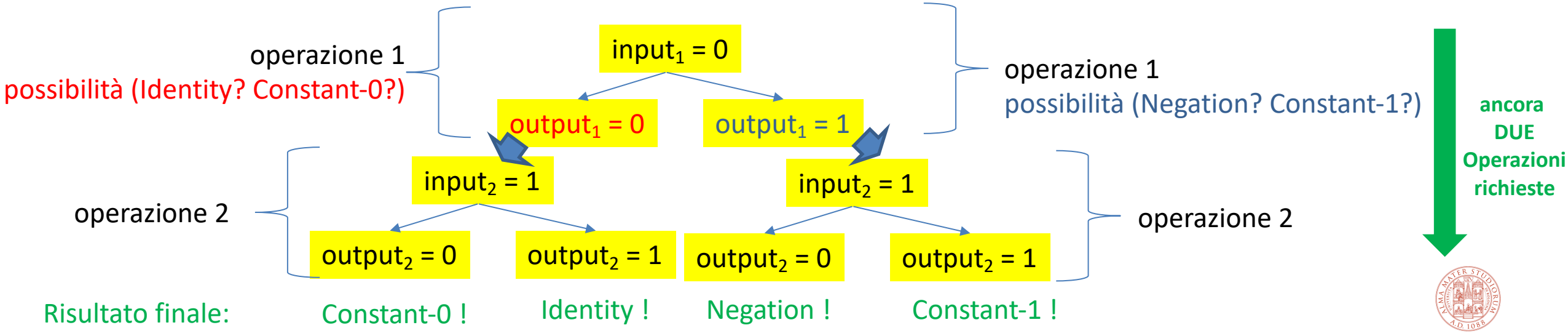
Problema: quante operazioni è necessario fare per sapere la risposta su un **Quantum Computer?** Possiamo solo provare a fornire valori di input e osservare Output per capire quale funzione sia.

The Deutsch Oracle problem

Immaginiamo che ci venga fornita una scatola nera che implementa una funzione su un bit.
Sappiamo che la funzione sarà una delle 4 possibili: Identity, Negation, Constant-0, Constant-1,
ma di quale si tratta?

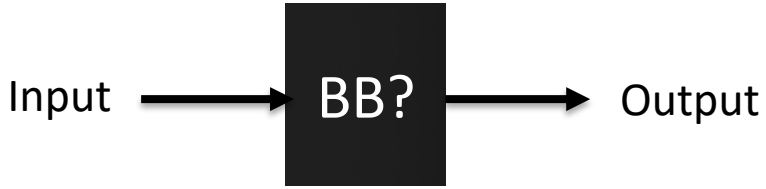


Problema: quante operazioni è necessario fare per sapere la risposta su un **Quantum Computer?**
Se usiamo un solo Qbit di input, sono sempre DUE operazioni quelle richieste.



The Deutsch Oracle problem

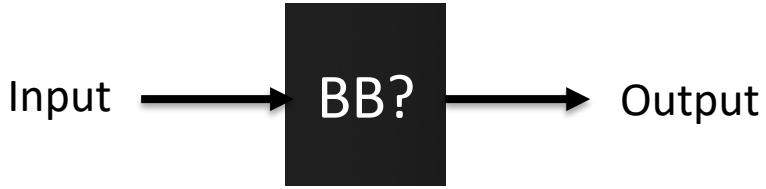
Immaginiamo che ci venga fornita una scatola nera che implementa una funzione su un bit.
Sappiamo che la funzione sarà una delle 4 possibili: Identity, Negation, Constant-0, Constant-1,
ma la NUOVA DOMANDA è: si tratta di una funzione costante o variabile?



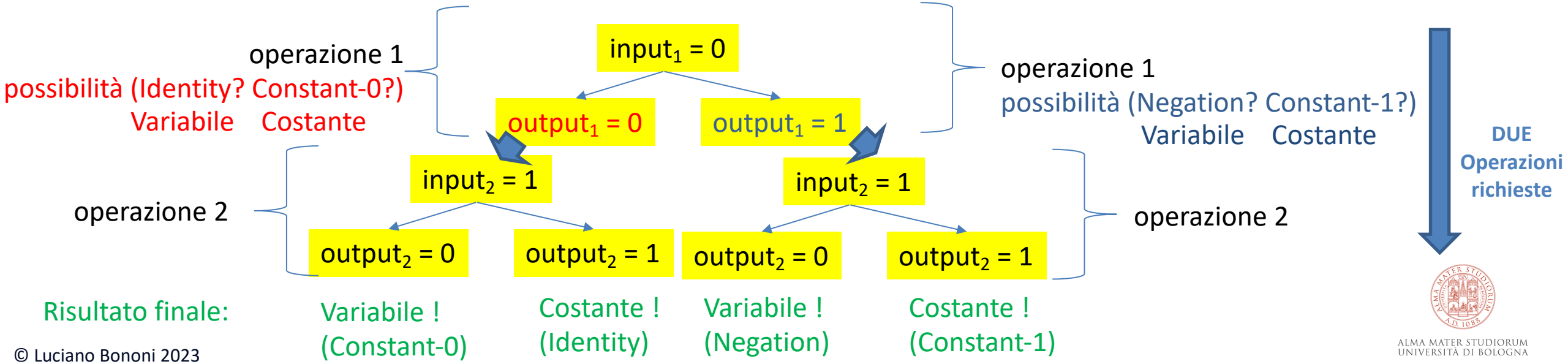
Problema: quante operazioni è necessario fare per sapere la risposta su un **Computer Classico?**

The Deutsch Oracle problem

Immaginiamo che ci venga fornita una scatola nera che implementa una funzione su un bit.
Sappiamo che la funzione sarà una delle 4 possibili: Identity, Negation, Constant-0, Constant-1,
ma la NUOVA DOMANDA è: si tratta di una funzione costante o variabile?

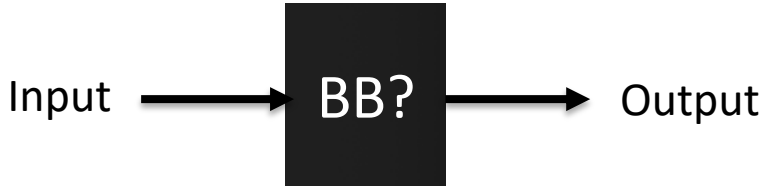


Problema: quante operazioni è necessario fare per sapere la risposta su un **Computer Classico?**



The Deutsch Oracle problem

Immaginiamo che ci venga fornita una scatola nera che implementa una funzione su un bit.
Sappiamo che la funzione sarà una delle 4 possibili: Identity, Negation, Constant-0, Constant-1,
ma la NUOVA DOMANDA è: si tratta di una funzione costante o variabile?



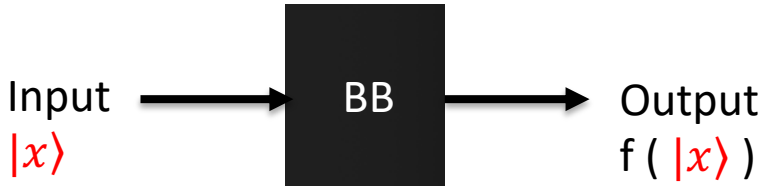
Problema: quante operazioni è necessario fare per sapere la risposta su un **Quantum Computer?**
Notare che la risposta è tra due possibilità e per distinguerle basta ora un solo Qbit!
Quindi basta fare **una sola operazione usando un Qbit in sovrapposizione!**
Dimezziamo il tempo di calcolo grazie al potere del Quantum Computing....

ma come si programma la domanda?

Dobbiamo definire come ognuna delle 4 funzioni possibili possa essere realizzata su un Quantum Computer...

The Deutsch Oracle problem

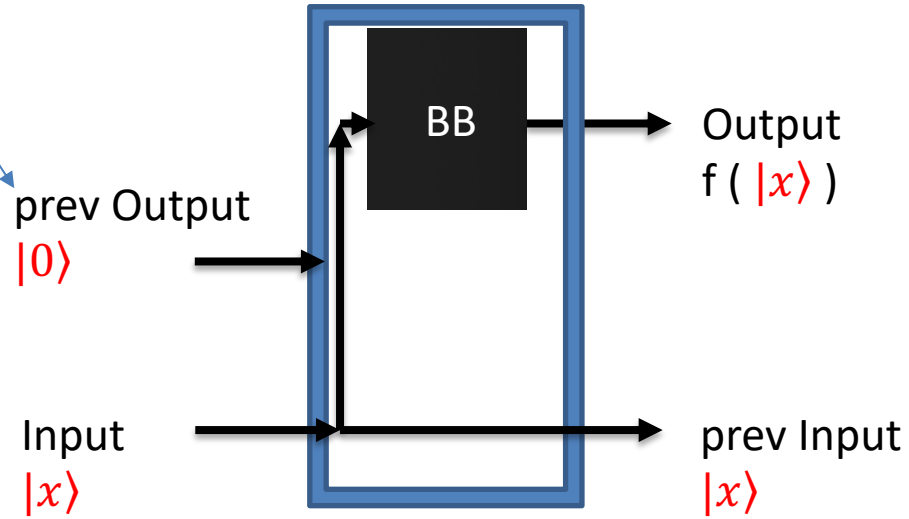
Problema: le funzioni Constant-0 e Constant-1 NON sono REVERSIBILI! (infatti sovrascrivono la storia)



Trucco: aggiungiamo un Qbit di output aggiuntivo solo per ricordare «da dove veniamo».

Dobbiamo quindi «ricablare» o riprogrammare il nostro circuito quantistico Black Box (BB) come segue:

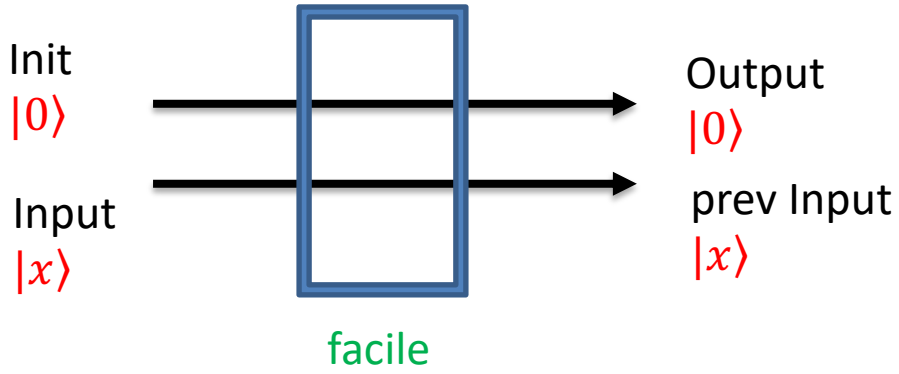
NB: questo «input» che aggiungiamo (chiamato prev Output) viene da noi sempre posto a $|0\rangle$, e quindi è solo tecnicamente un input della rete.
In realtà è un input costante.
E' come definire una costante in un programma, quindi non è un input aggiuntivo del problema.



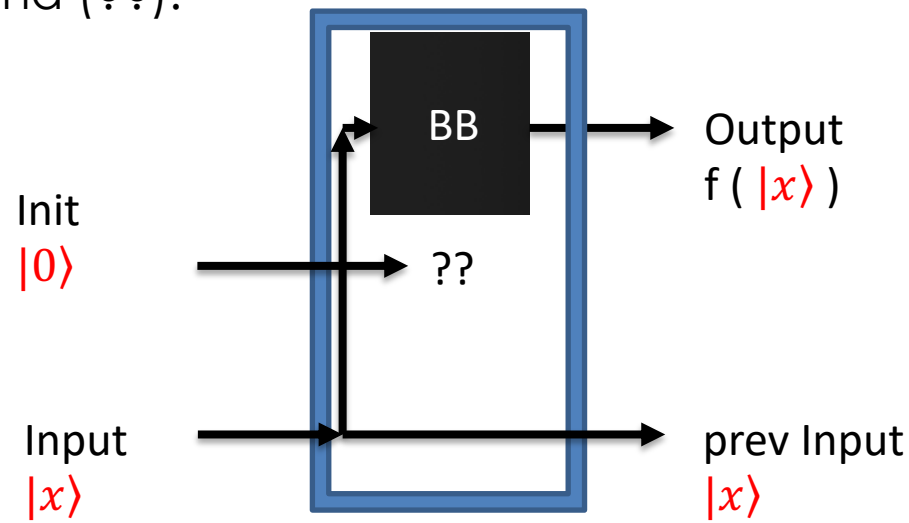
The Deutsch Oracle problem

Vediamo come ricablare le due funzioni Constant-0 e Constant-1 perchè producano il modello di funzione generale a destra (a due Qbit) risolvendo la forma interna (??):

circuito Constant-0

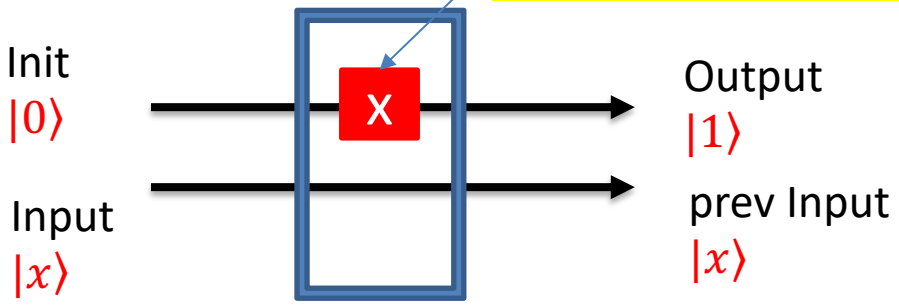


facile



circuito Constant-1

NB: sappiamo fare circuito BitFlip (x) di un Qbit

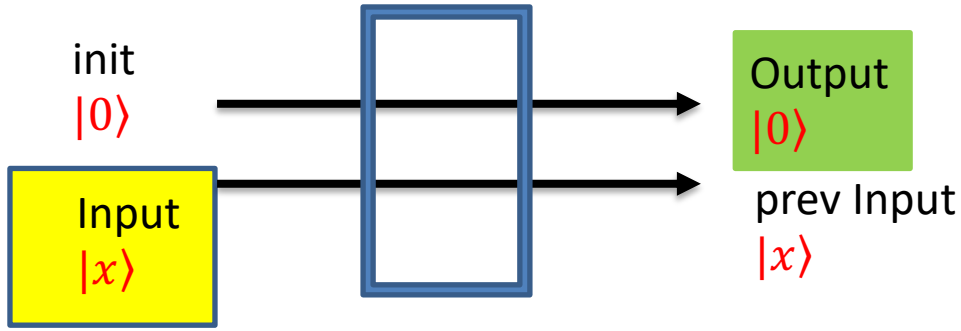


facile usando X internamente

The Deutsch Oracle problem

Ora vediamo se le due funzioni Constant-0 e Constant-1 così ricablate sono quindi reversibili?

circuito Constant-0



Vediamo i due casi possibili con Constant-0:

Se Output = $|0\rangle$ e prev Input = $|0\rangle$
allora gli input erano Init $|0\rangle$

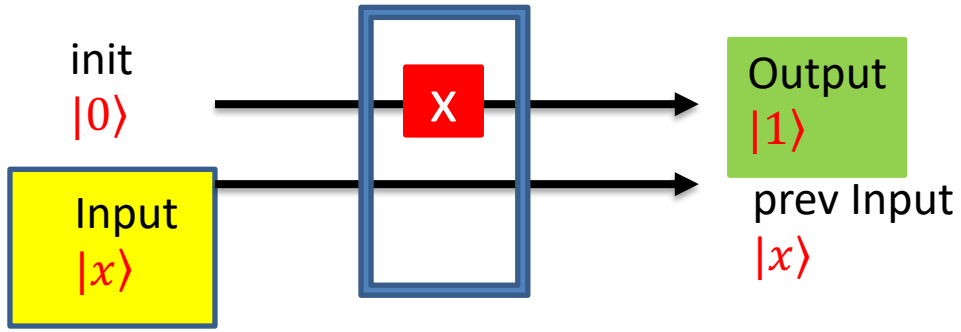
Se Output = $|0\rangle$ e prev Input = $|1\rangle$
allora gli input erano Init $|0\rangle$

e Input $|0\rangle$

e Input $|1\rangle$

OK!
Computazione
REVERSIBILE
malgrado
OUTPUT sia
costante a $|0\rangle$

circuito Constant-1



Vediamo i due casi possibili con Constant-1:

Se Output = $|1\rangle$ e prev Input = $|0\rangle$
allora gli input erano Init $|0\rangle$

Se Output = $|1\rangle$ e prev Input = $|1\rangle$
allora gli input erano Init $|0\rangle$

e Input $|0\rangle$

e Input $|1\rangle$

OK!
Computazione
REVERSIBILE
malgrado
OUTPUT sia
costante a $|1\rangle$

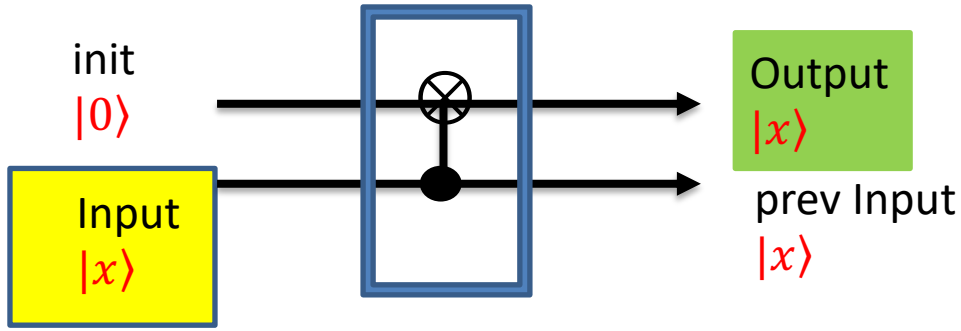
The Deutsch Oracle problem

Ora creiamo il circuito interno Quantum Computing anche per la funzione Identity e Negation (già reversibili)

C

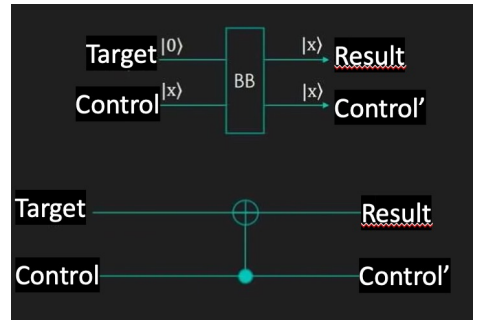
NB: quindi questo circuito interno per Identity è in realtà un CNOT tra i due input

circuito Identity

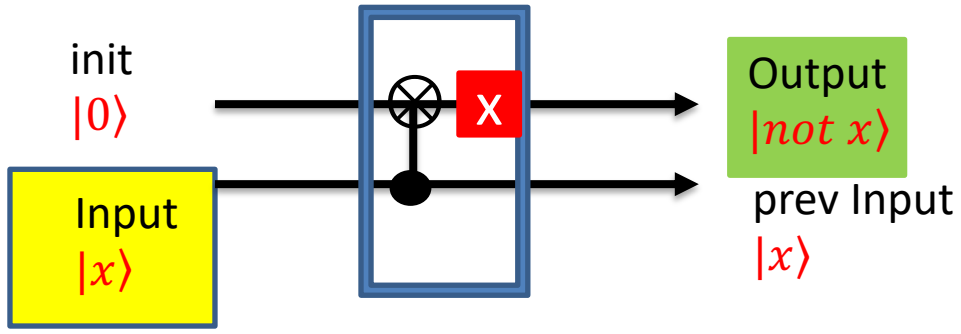


Se Input (Control) è $|0\rangle$ (e init (Target) è $|0\rangle$) allora Output (Result) è $|0\rangle$ e prev Input (Control') è $|0\rangle$

Se Input (Control) è $|1\rangle$ (e init (Target) è $|0\rangle$) allora Output (Result) è $|1\rangle$ e prev Input (Control') è $|1\rangle$



circuito Negation



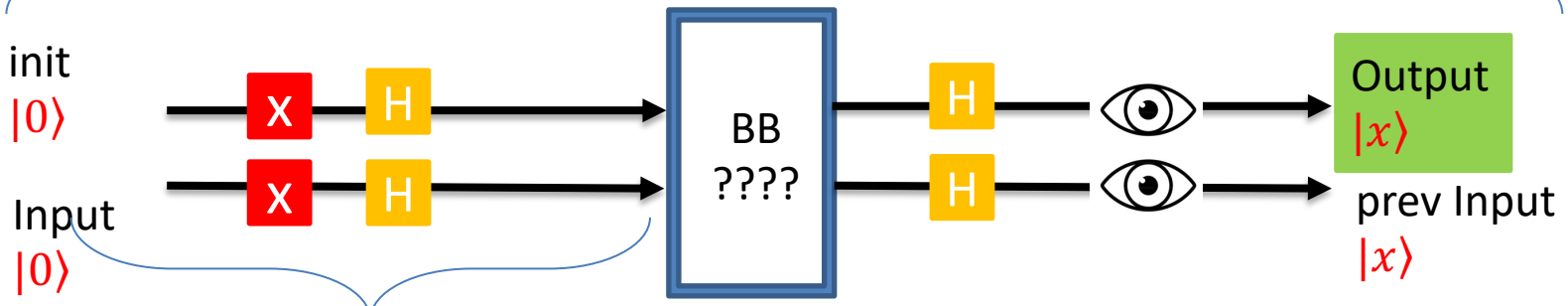
banalmente è il circuito Identity precedente (CNOT) con uscita negata (da un circuito flip bit)

C X

The Deutsch Oracle problem

Ora, se queste sono le 4 funzioni (reversibili) possibili che potremmo trovare implementate nel Black Box, come programmiamo la soluzione che ci dice se la funzione BB sia costante o variabile in un solo passo sul Quantum Computer?

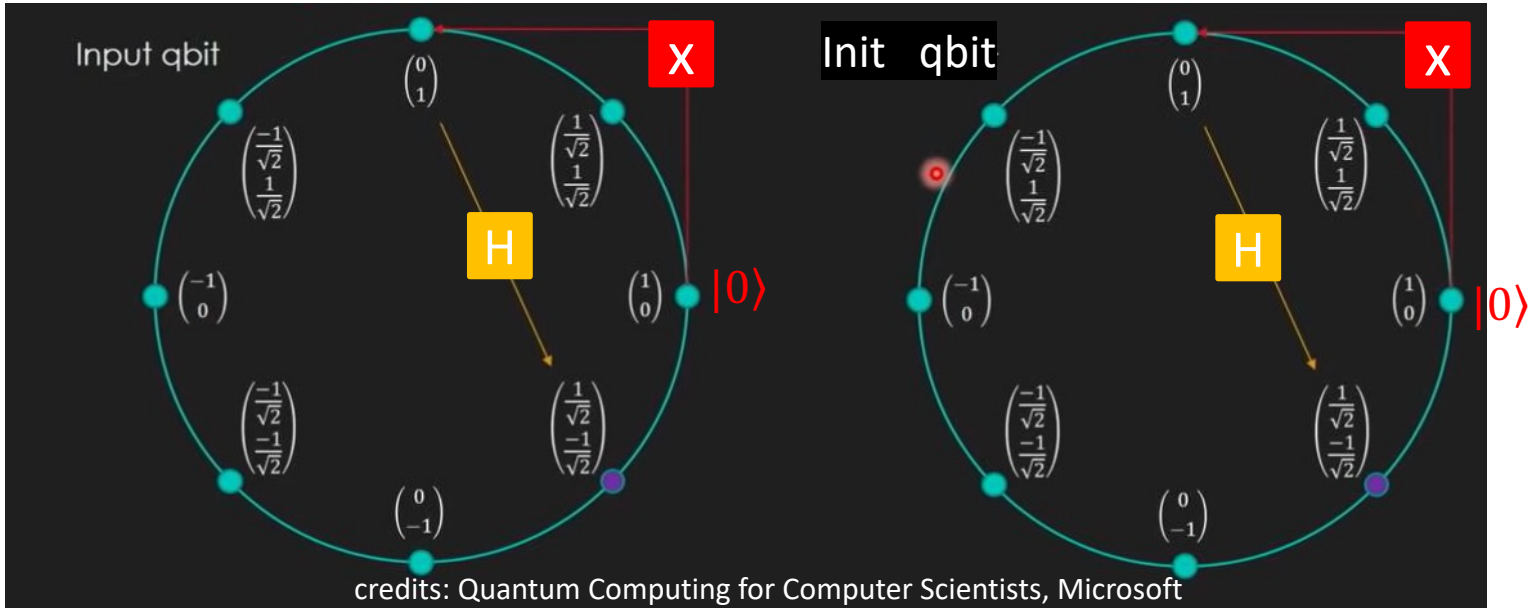
Programma Quantistico che risolve la domanda: BB è una funzione costante o variabile?



Vediamo prima il pre-processing.

Arriviamo nello stato

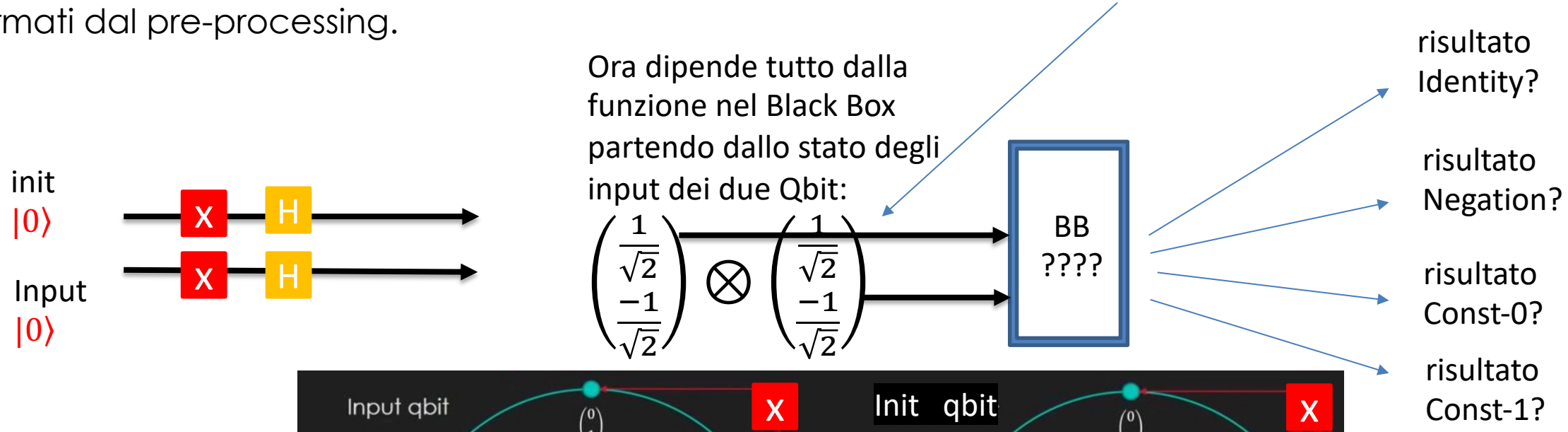
$$\begin{pmatrix} 1 \\ \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} 1 \\ \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}$$

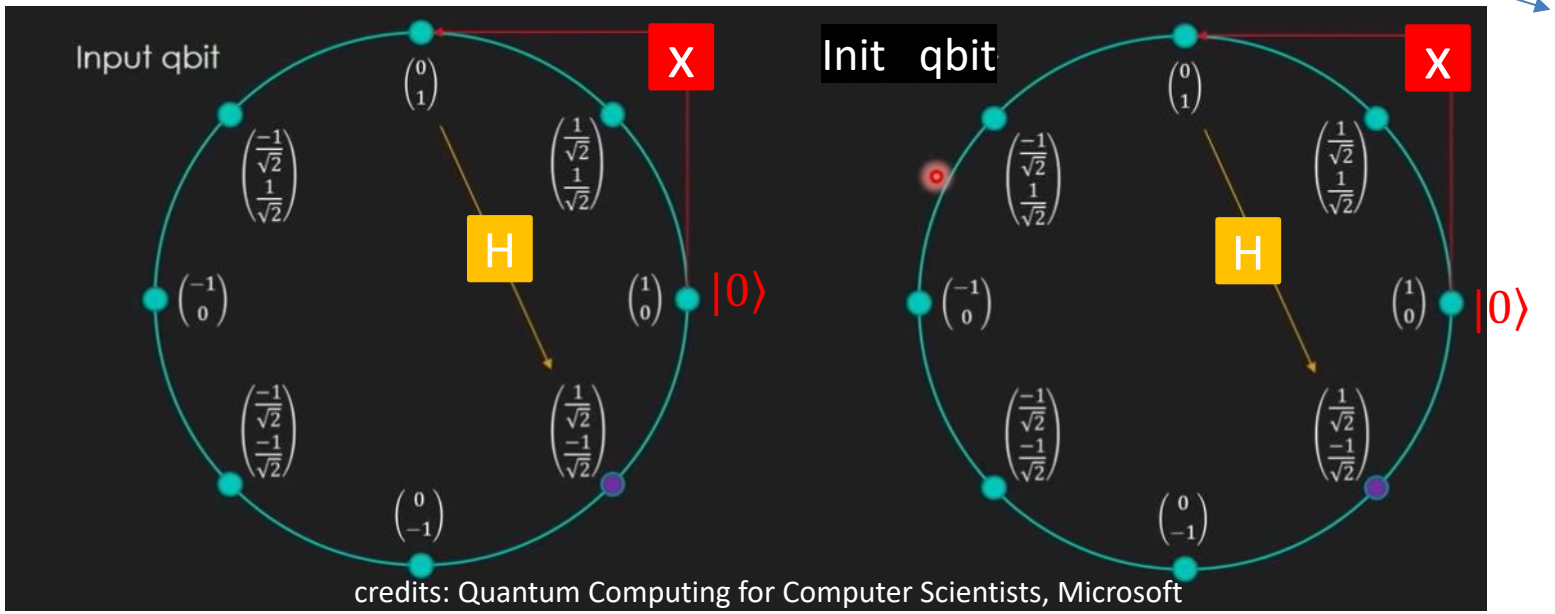


credits: Quantum Computing for Computer Scientists, Microsoft

The Deutsch Oracle problem

Ora, vediamo come la funzione BB modifica in tutti i 4 modi possibili gli stati dei Qbit di input trasformati dal pre-processing.





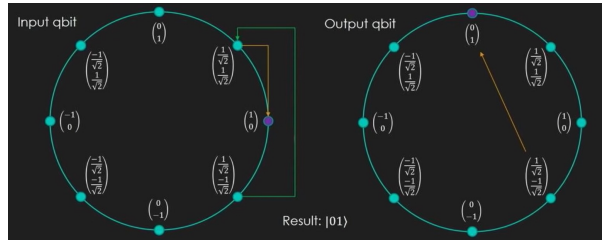
credits: Quantum Computing for Computer Scientists, Microsoft

The Deutsch Oracle problem

Vediamo i risultati dello State Vector per le 4 funzioni possibili a partire dall'input dell'operazione:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} = \overset{\text{CNOT}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}} \begin{bmatrix} 1/2 \\ -1/2 \\ -1/2 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 1/2 \\ -1/2 \\ 1/2 \\ -1/2 \end{bmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}$$

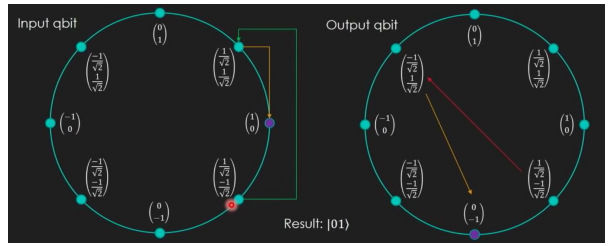
H = $|01\rangle$



BB Identity (made with CNOT)

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1/2 \\ -1/2 \\ -1/2 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 1/2 \\ -1/2 \\ -1/2 \\ 1/2 \end{bmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}$$

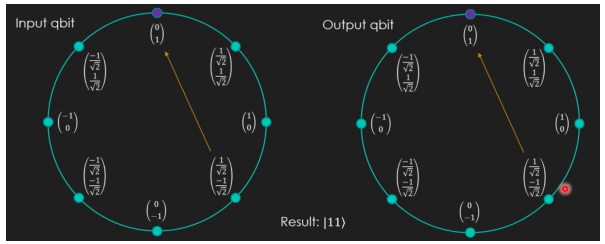
H = $|01\rangle$



BB Negation

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} = \cdots\cdots$$

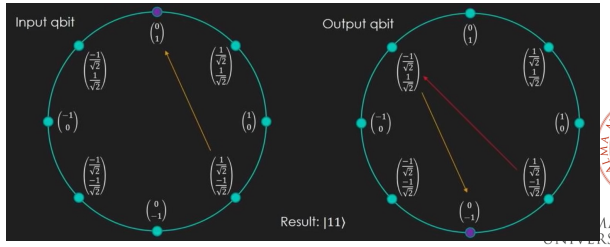
H = $|11\rangle$



BB Const-0

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} = \cdots\cdots$$

H = $|11\rangle$



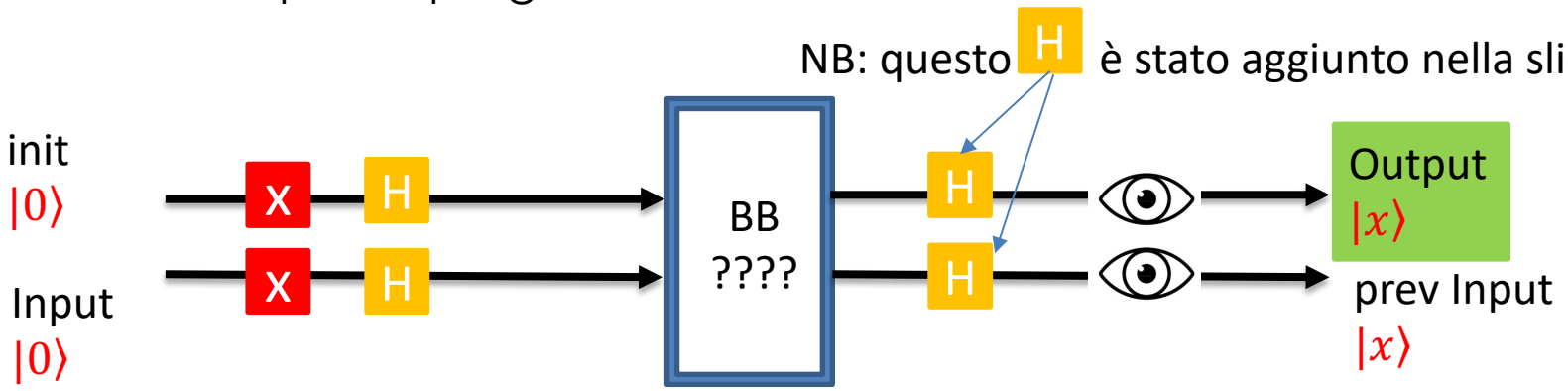
BB Const-1

notare che qui c'è stato anche il bit flip rispetto alla riga sopra (in quanto negation è identity più un bitflip finale

notare che qui in uscita avremo H (non mostrato algebricamente), e poi leggiamo i Qbits indicati in rosso.

The Deutsch Oracle problem

Ora, osserviamo che dato questo programma sul BlackBox centrale:



Se BB = funzione costante (Constant-0 o Constant-1) immettendo Init e Input $|00\rangle$ misureremo Output e Prev Input = $|11\rangle$

Se BB = funzione variabile (Identity o Negation) immettendo Init e Input $|00\rangle$ misureremo Output e Prev Input = $|01\rangle$

Risultato NOTEVOLE!

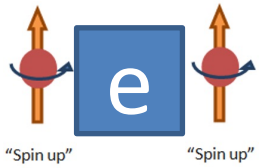
risposta corretta ottenuta dimezzando gli step di esecuzione rispetto al calcolo non quantistico

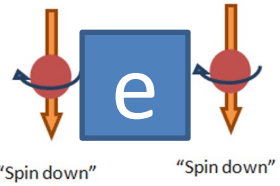
… ma non è ancora tutto

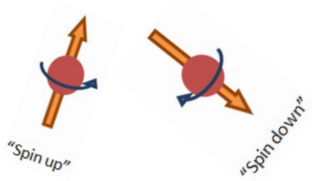
Perchè esiste anche **l'entanglement tra Qbits**. e un nuovo operatore che fa cose stupende.

faccio l'entaglement di 2 Qbit vicini qui e adesso (senza osservarli)

I Qbit rimangono «impigliati» tra loro.



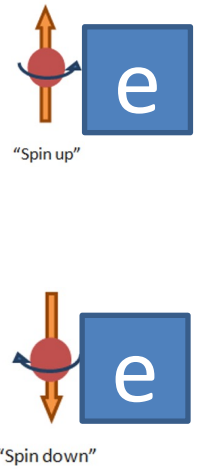




… ma non è ancora tutto

Perchè esiste anche **l'entanglement tra Qbits**. e

trasporto uno dei due Qbit a distanza enorme
(mettendoci il tempo che serve al viaggio)
senza mai osservarli



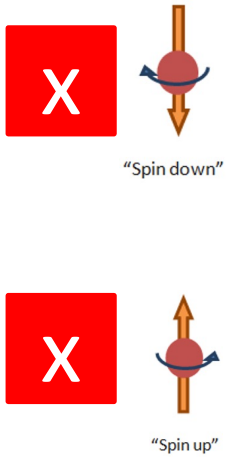
distanza di anni luce

… ma non è ancora tutto

Perchè esiste anche **l'entanglement tra Qbits**.

e

ora decido di modificare il valore del Qbit
rimasto qui sulla terra al tempo T

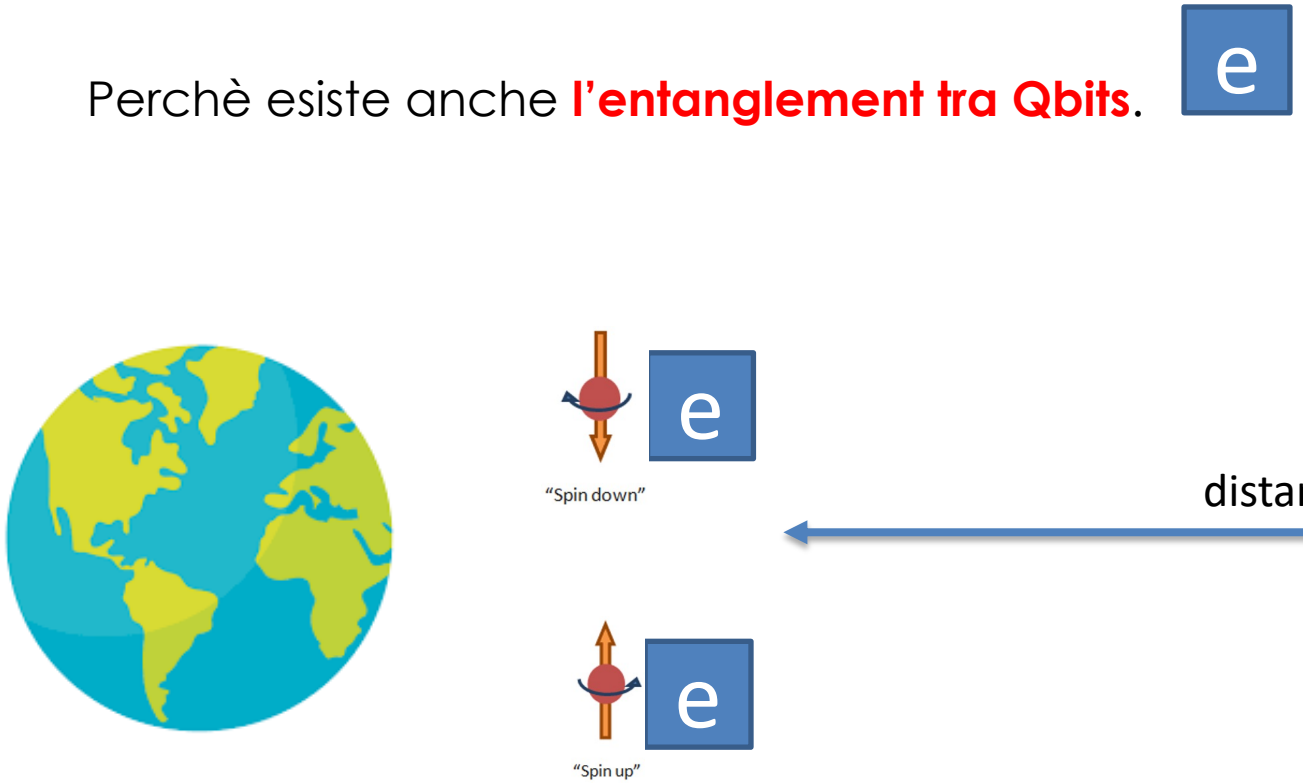


distanza di anni luce

Qbit entagled a quelli rimasti
sulla terra
(non ancora osservati)

… ma non è ancora tutto

Perchè esiste anche **l'entanglement tra Qbits**.

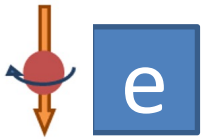


se dal tempo T in poi osservo i Qbit lontani,
«istantaneamente»,
troverò lo stesso valore sul Qbit modificato sulle terra

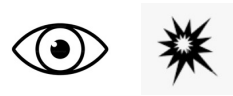
… ma non è ancora tutto

Perchè esiste anche **l'entanglement tra Qbits**.

se dal tempo T in poi osservo i Qbit lontani,
«istantaneamente»,
troverò lo stesso valore sul Qbit modificato sulle terra,
ma avrò consumato il Qbit lontano.

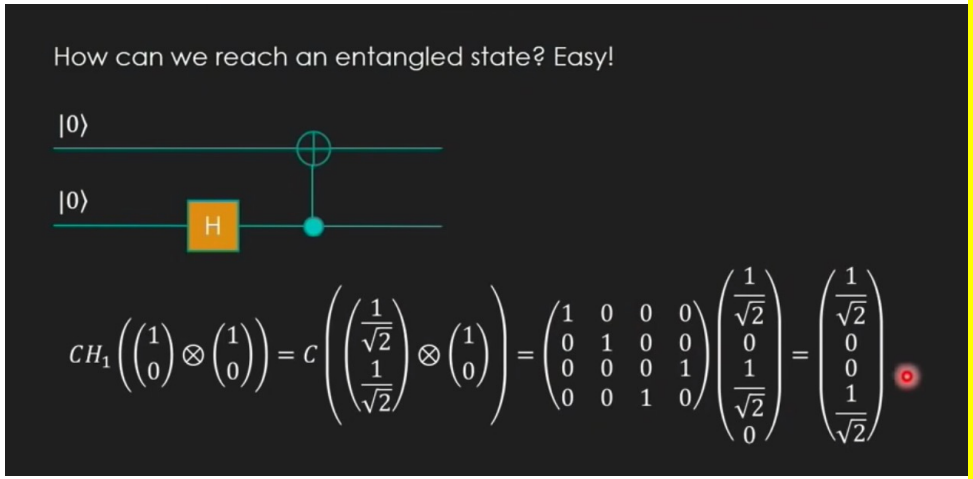


distanza di anni luce



Potenza dell'entanglement?

Ma quindi possiamo anche coordinare due sistemi a distanza enorme in tempo (quasi) nullo?
Si, grazie all'entanglement! Ma è possibile fare una porta di entaglement tra Qbits? Eccola!



Circuito di Entaglement di due Qbits realizzato con una porta H e una CNOT. Si noti che mettendo in Entanglement due Qbit di input $|0\rangle$ e $|0\rangle$ la porta H porta il primo in sovrapposizione (s = 0 e 1 al 50%), quindi avremo una situazione di Qbit $|s\ 0\rangle$ (con vettore di stato che prevede due casi possibili al 50%: $|s\ 0\rangle$ o $|s\ 1\rangle$. Tale situazione viene passata al CNOT che genera il vettore di stati finale a destra in figura (punto rosso), che prevede solo due casi possibili: 50% $|0\ 0\rangle$ e 50% $|1\ 1\rangle$. Questo significa che se cambio valore a un Qbit, anche l'altro assume lo stesso stato e non possono esistere i due Qbit in stati diversi tra loro.... e ciò vale anche a grande distanza in tempo più veloce della luce (come osservato sperimentalmente).
Ma non chiedetemi perchè... ☺ (se lo scoprirete vincerete il Nobel)

Avremo presto Co-Processori Quantistici che affiancano Processori Tradizionali? probabile.
Avremo presto «reti di comunicazione quantistiche» che sincronizzano sistemi in tempo nullo? probabile (NB: NON significa che comunichino... ma pensate a come potremmo fare la comunicazione).

Conclusione

Al mondo ci sono $\binom{0}{1} \otimes \binom{1}{0}$ gruppi di informatici...

Voi a quale dei due gruppi appartenete?

Conclusione

Al mondo ci sono $\binom{0}{1} \otimes \binom{1}{0}$ gruppi di informatici...

$\binom{0}{1} \otimes \binom{1}{0} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ = state product indica un valore al 100% di probabilità uguale a 2

1) quelli che non capiscono il Quantum Computing
2) quelli che credono erroneamente di averlo capito.

Voi a quale dei due gruppi appartenete?

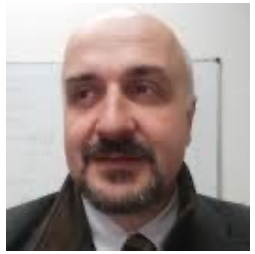
Conclusione

Al mondo ci sono $\binom{0}{1} \otimes \binom{1}{0}$ gruppi di informatici...

$\binom{0}{1} \otimes \binom{1}{0} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ = state product indica un valore al 100% di probabilità uguale a 2

1) quelli che non capiscono il Quantum Computing

✅ 2) quelli che credono erroneamente di averlo capito.



Voi a quale dei due gruppi appartenete?

Conclusione

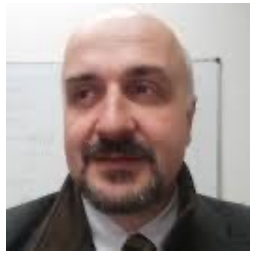
Al mondo ci sono $\binom{0}{1} \otimes \binom{1}{0}$ gruppi di informatici...

$\binom{0}{1} \otimes \binom{1}{0} =$ [ ]

un valore al 100% di 2

1) quelli tum Computing

✅ 2) quelli di averlo capito.



Voi a quale dei due gruppi appartenete?

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Luciano Bononi

Dipartimento di Informatica – Scienza e Ingegneria

luciano.bononi@unibo.it

https://www.unibo.it/sitoweb/luciano.bononi