Tullio Levi-Civita

Università degli Studi di Padova

Scuola di scienze

Corso di Laurea in
Informatica

Analisi della complessità dei circuiti quantistici

Relatore:
Prof. Alessandro Brighente

Laureando:
Alberto Casado Moreno
2000550

Anno Accademico 2022/2023

Abstract

Una delle differenze principali dei computer quantistici rispetto ai computer classici è l'utilizzo di circuiti e logica reversibile. Questi circuiti riducono la dissipazione di energia e permettono l'implementazione di algoritmi quantistici, algoritmi che presentano una complessità computazionale inferiore rispetto alla controparte classica. Questa tesi si propone di esaminare in dettaglio il funzionamento dei qubit e delle porte logiche quantistiche. Verranno analizzate le proprietà fisiche dei qubits, come l'entanglement e la sovrapposizione di stati, e come queste possano essere sfruttate nei circuiti quantistici. Verrà inoltre analizzata una nuova porta logica e come questa potrebbe ridurre il costo di alcuni algoritmi.

La prima parte della tesi sarà dedicata alla descrizione delle basi teoriche della computazione quantistica. Verranno descritti i qubits, le loro proprietà fisiche, la loro implementazione e rappresentazione. Verrà descritto il funzionamento delle porte logiche quantiche, confrontandole con la controparte classica, come queste permettono di manipolare i qubits e sfruttarne le proprietà fisiche. Seguiranno una serie di porte logiche quantiche, fondamentali per la maggior parte di algoritmi, illustrandone la rappresentazione e descrivendone i parametri.

Nella seconda parte verrà presentato una nuova porta logica, mostrando funzionamento e proprietà.Verrà mostrato come questa porta potrebbe essere utilizzata per ridurre la complessità dei circuiti di addizione di qubit. Verranno presentati i circuiti creati utilizzati questa porta logica e ne verranno studiati i parametri.

Indice

Capitolo 1

Introduzione

La computazione quantica è la disciplina che sfrutta le proprietà quantiche della materia per costruire calcolatori capaci di utilizzare queste propietà per svolgere calcoli.

A partire dal secolo scorso, la computazione quantica è stata studiata e teorizzata, ma è ancora in una fase di sviluppo attivo. I computer quantistici possono essere utilizzati per creare algoritmi con complessità inferiore rispetto ai computer classici, questo è possibile grazie ai qubit e porte logiche quantiche. La riduzione della complessità degli algoritmi beneficia molti rami dell'inoformatica e matematica, come la crittografia e l'intelligenza artificiale.

Purtroppo però la costruzione su larga scala di computer quantistici è una sfida tecnologica data dai materiali richiesti e dalle basse temperature che il calcolatore necessita per funzionare.

1.1 Contesto

Con il progressivo ridimensionamento della tecnologia, aumenta l'integrazione di circuiti complessi nei circuiti integrati. Nonostante i progressi nella progettazione di circuiti di integrazione su larga scala, la dissipazione di potenza rimane una delle questioni più importanti nella progettazione di circuiti digitali. Ogni calcolo logico irreversibile utilizza $KTln2$ joule di energia come perdita di dati per ogni pezzo di informazione, come dimostrato da Landauer [1] nel 1961. La costante di Boltzmann è $K = 1,38 \times 10^{-23}\ m^2kg^2k^{-1}$ (joule kelvin^{-1}), e T è la temperatura assoluta utilizzata per il calcolo.
Bennet [2] ha dimostrato che la perdita di energia non si verificherebbe se un

circuito digitale avesse una progettazione logica reversibile e un'implementazione fisica reversibile.

Oggi, la logica reversibile sta suscitando un grande interesse tra i progettisti di circuiti perché consuma meno energia e, inoltre, il computer quantistico si basa sulla logica reversibile [3]. L'implementazione di circuiti reversibili, oltre al computer quantistico, è stata esplorata in applicazioni come l'elaborazione di segnali digitali (DSP), il DNA, la grafica computerizzata, la nanotecnologia, la crittografia e la progettazione VLSI ad alta velocità.
Circuiti reversibili, come `Half Adder` (HA) e `Full Adder` (FA) sono stati proposti di recente per una varietà di applicazioni. A causa dell'ampio utilizzo dei circuiti moltiplicatori nella `Arithmetic Logic Unite` (ALU) moderna e nei sistemi informatici, i circuiti moltiplicatori rivestono una particolare importanza tra questi circuiti reversibili e la progettazione di un moltiplicatore con buone prestazioni è quindi importante [4].

1.2 Conoscenze Preliminari

Logica reversibile

Alla base della computazione quantistica vi è la logica reversibile. Un circuito è reversibile se:

- Presenta un numero uguale di input ed output;
- L'output contiene informazioni sufficienti per ricostruire l'input, ossia non è presente perdita di informazione degli input, il circuito può essere "invertito" [5].
- Ciascuna configurazione di input provoca una configurazione di output distinta, ovvero ogni input è mappato ad un output unico e distinto.

Gate Reversibili e Quantistici

I circuiti classici utilizzano segnali elettrici per rappresentare l'informazione, ovvero i bit, segnali che vengono manipolati utilizzando le porte logiche. Mentre i circuiti quantici utilizzano un'unità di informazione chiamata qubit, per codificare l'informazione. Le porte logiche quantistiche sono intrinsecamente reversibili [3].

Le porte logiche quantiche si possono separare in due gruppi: le porte elementari e quelle complesse. Le porte elementari sono porte che agiscono solo su uno o due qubits, mentre quelle complesse agiscono su più qubit, inoltre queste porte possono essere ricavate tramite la composizione di porte logiche elementari.

1.3 Struttura della Tesi

La tesi si può dividere in due parti, la prima serve per introdurre i qubit e le porte quantiche, mentre nella seconda parte si analizzerà una porta particolare e la sua implementazione nei circuiti di somma.

Si comincerà con l'introduzione dei qubit, come sono stati implementati, la loro rappresentazione e le loro proprietà. Si continuerà con le porte quantiche, che permettono di sfruttare le proprietà dei qubit. Dopo averne analizzato la rappresentazione e i parametri si analizzeranno alcune delle porte più importanti per la costruzione di circuiti.

Nella seconda parte si analizzerà la porta $\sqrt{V}$ sottolineandone le proprietà, per poi analizzare circuiti di somma creati a partire da essa. Infine si vedrà un esempio di implementazione dei circuiti visti.

Capitolo 2

Qubits

L'unità base di un computer quantistico sono i qubit, acronimo di "quantum bit", che rappresenta la base dell'informazione della computazione quantistica. Il qubit non è semplicemente l'analogo ad un bit classico, esso rappresenta e riporta le proprietà quantiche delle particelle come la sovrapposizione, l'entanglement ed il teletrasporto. Queste proprietà permettono di utilizzare i qubit come più che semplici bit, il cui valore può essere rappresentato da un 1 e da uno 0. Infatti i qubit vengono rappresentati da una coppia di numeri complessi che permettono di descrivere la sovrapposizione degli stati $|0\rangle$ e $|1\rangle$, non solo, ma tramite apposite operazioni, è possibile collegare qubits tramite la proprietà fisica dell'entanglement.

2.1 Implementazione

Inizialmente per la rappresentazione Hardware dei qubits si pensava di utilizzare gli elettroni di un atomo, in quanto possedevano già le proprietà quantiche che si volevano sfruttare. Tuttavia, si è presto scoperto che collegare e controllare un gran numero di elettroni di atomi diversi rappresentava una sfida tecnica considerevole. Di conseguenza, gli sforzi si sono spostati verso la creazione di "atomi artificiali".

Attualmente il metodo più diffuso ed utilizzato per la rappresentazione dei qubit è il Transmon, utilizzato anche da Google e IBM. Alla base dell'implementazione del Transmon vi è la giunzione Josephson che è costituita da due strati di materiale superconduttore separati da una sottile barriera isolante. Per leggere e controllare i qubitsviene utilizzato un dispositivo chiamato SQUID (Superconducting Quantum Interference Device). Questo tipo di qubit ha dimostrato essere

particolarmente adatto per l'elaborazione quantistica grazie alla facilità e stabilità con cui consente di controllare e manipolare i due stati quantistici fondamentali. Due parametri da considerare quando si valutano modelli hardware di qubits sono il tempo di coerenza e la scalabilità. Il tempo di coerenza rappresenta quanto a lungo un qubit può mantenere il suo stato quantico prima di perdere affidabilità a causa dell'interazione con l'ambiente circostante. La scalabilità, d'altra parte, è l'abilità di creare e controllare più qubit in modo simultaneo, questa proprietà limita il quantitativo di qubit che un computer può utilizzare, attualmente intorno a qualche centinaio. Migliorare questi parametri permette di creare computer quantici capaci di eseguire algoritmi sempre più complessi.

2.2 Rappresentazione

I bit classici hanno solo due possibili stati 0 ed 1, i qubits invece posso essere, oltre che in quei due stati, anche in una coombinazione lineare dei due. Questo è possibile grazie alla proprietà fisica della sovrapposizione. Per poter rappresentare questo stato di sovrapposizione i qubits vengono rappresentati utilizzando due numeri complessi, con le seguenti notazioni:

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle = \begin{pmatrix} a \\ b \end{pmatrix},$$

dove $|a|^2$ e $|b|^2$ sono, rispettivamente, le probabilità di osservare il qubit nello stato 0 oppure 1 nel momento della misurazione. La prima notazione si chiama **bra-ket notation**, spesso utilizzata in mecchanica quantica per descrivere uno stato quantico. Mentre la seconda è la forma matriciale, piu utile ed utilizzata per calcolare l'evoluzione di un sitema a seguito dell'applicazione di porte logiche. Si possono usare entrambe le notazioni per rappresentare sistemi con più di un qubit.

$$|\psi\rangle\otimes|\varphi\rangle = (a\,|0\rangle + b\,|1\rangle)\otimes(c\,|0\rangle + d\,|1\rangle) = ac\,|00\rangle + ad\,|01\rangle + bc\,|10\rangle + bd\,|11\rangle = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}.$$

Bloch sphere

La sfera Bloch è un modo di visualizzare lo stato di un qubit tramite una sfera, La sfera di Bloch è utile per visualizzare in modo chiaro le rotazioni rispetto agli assi dati dall'applicazione di qualche porta quantica, come la porta `Phase Shift`.

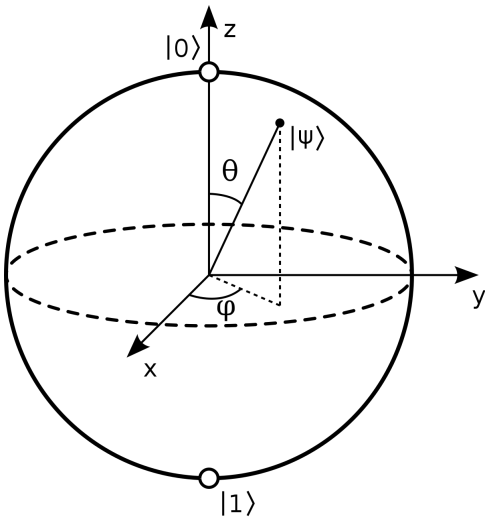


Figura 2.1: Rappresentazione della sfera di Bloch.

Il qubit viene rappresentato da un vettore, il suo stato è il punto della sfera a cui il vettore è direzionato. In alto lo stato $|0\rangle$ e in basso lo stato $|1\rangle$, tutti gli altri punti della sfera sono stati di sovrapposizione. In particolare, l'equatore rappresenta gli stati in cui $|a|^2 = |b|^2 = 0.5$.

2.3 Proprietà

I qubit presentano varie probità derivante dalla loro natura quantica.

Superposition

La sovrapposizione di stati è una proprietà fondamentale per i qubit ed è ciò che gli permetti di avere più di due semplici stati. Come visto nella sezione precedente il qubit può essere in una combinazione lineare degli stati $|0\rangle$ e $|1\rangle$. In termini fisici questo stato si chiama *superposition*, o sovrapposizione in italiano, ovvero la capacità di un sistema quantistico di trovarsi in più stati contemporaneamente fino a quando non viene misurato.

Misurazione

La sovrapposizione permette a un qubit di trovarsi in un numero indefinito di stati. Questo potrebbe portare a pensare alla possibilità di immagazzinare una quantità infinita di dati su un singolo qubit, in particolare sulle sue componenti `a` e `b`. Purtroppo questo non è possibile in quanto non è possibile misurare il valore delle componenti `a` e `b` di un qubit.

Quando un qubit viene misurato esso collassa in uno stato classico e perde le propietà quantiche, ossia il risultato di un'operazione di misura su un qubit darà come risultato un normale bit. Se il qubit misurato è definito come $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$ il sistema avrà probabilità $|a|^2$ di collassare (quindi essere misurato) nello stato $|0\rangle$ e probabilità $1 - |a|^2$ ossia $|b|^2$ di collassare nello stato $|1\rangle$

Entaglement

L'*entaglement*, o intreccio in italiano, è il fenomeno quantico che si verifica quando un gruppo di particelle entrano in una relazione tale che lo stato quantistico di ogni particella del gruppo non può più essere descritto in modo indipendente dallo stato delle altre, anche quando le particelle sono separate da una grande distanza. Applicato ai qubit si traduce in qubit che non possono più essere considerati singolarmente ma devono essere essere utilizzati considerando il sistema di cui fanno parte. Lo stato di intreccio fra qubit avviene, ad esempio, quando si applica una porta logica che agisce su più qubit mente questi sono in uno stato di sovrapposizione. La pare di intreccio tra qubit verrà approfondita maggiormente nella sezione *Porte Logiche Importanti* (3.3) dedicata alle porte logiche, visto che è grazie a queste ultime che si possono creare questi stati.

Capitolo 3

Quantum Gates

Le porte logiche quantistiche sono circuiti che operano su uno o più qubit, manipolandone lo stato.

A differenza delle porte logiche classiche, le porte logiche quantiche sono reversibili, ciò vuol dire che ogni operazione eseguita su un qubit può essere invertita senza perdita di informazione (ad eccezione della porta `NOT` che non presenta perdita di informazione anche nella controparte classica). Tutte le porte logiche classiche possono essere replicate con un apposito circuito quantistico; per mantenere la reversibilità questo circuito avrà, oltre all'output desiderato, un numero addizionale di qubit, necessario per eguagliare il numero di input.

Analogamente ai computer classici, le porte logiche quantiche servono per svolgere operazioni sui qubits. In aggiunta alle operazioni classiche, come `AND`, `XOR` e `NOT`, è possibile svolgere anche operazioni che sfruttino la natura quantica dei qubit, ad esempio mettendo in sovrapposizione di stati un qubit oppure collegando due o più qubits tramite *Entaglement*.

3.1 Parametri

Ci sono diversi parametri per valutare e confrontare i circuiti quantistici. Questi parametri includono il costo quantistico, gli ingressi costanti, il numero di porte, la profondità del circuito, il conteggio dei qubit e l'output di scarto. Ottimizzare questi parametri è importante per accelerare i circuiti e ridurre il consumo di energia.

Quantum Cost

Il **Quantum Cost** (QC) di un circuito è una metrica utilizzata per quantificare le risorse necessarie per implementare il circuito. Il costo quantistico di un circuito è definito dal numero totale di porte quantistiche elementarie per la sua implementazione. È preferibile ridurre il costo quantistico perché ciò porta a un circuito più efficiente, che richiede meno risorse hardware e, nella maggior parte dei casi, meno tempo di esecuzione.

Constant Input

Il **Constant Input** (CI) di un circuito è il numero di qubit aggiuntivi necessari per mantenere la reversibilità di un circuito quantistico. Ad esempio, una `Porta di Peres` può essere utilizzata come un Half-Adder, che accetta in input i due qubit da sommare ed un terzo ingresso come CI di $|0\rangle$. Poiché gli input costanti sono qubit aggiuntivi, è importante cercare di minimizzare questa metrica il più possibile.

Garbage Output

Il **Garbage Output** (GO) di un circuito è il numero di qubit di output che non sono i qubit che contengo l'informazione dell'operazione svolta dal circuito. I qubit GO sono necessari per mantenere la reversibilità di un circuito quantistico. Ad esempio, una `Porta di Peres` utilizzata come Half-Adder produrrà due qubit che contengono le informazioni desiderate (la somma ed il resto) ed un qubit di GO.

3.2 Rappresentazione

Per rappresentare il funzionamento di una porta logia classica si utilizzano le tabelle logiche che permettono di rappresentare tutte le possibili combinazioni di input con il corrispettivo output. Ad esempio la seguente è la tabella di verità della porta `XOR`:

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Tabella 3.1: Tabella di verità della porta logica XOR.

Non è possibile costruire la tabella di verità per una porta logica quantica, per lo meno non è possibile farlo con tutti i possibili input.

Le porte logiche quantiche vengono rappresentate da matrici unitarie, ovvero una matrice quadrata complessa invertibile, la cui inversa è uguale alla sua coniugata trasposta. Ovvero A è una matrice unitaria se $AA^\dagger = A^\dagger A = I$. La dimensione della matrice è proporzionale al numero di input della porta, una porta con n input è rappresentata da una matrice unitaria di dimensione 2^n.

3.3 Porte logiche importanti

Essendo rappresentate da matrici unitarie, si ha a disposizione una notevole quantità di porte logiche quantiche.

Esistono infinite porte logiche quantiche, alcune di queste giocano un ruolo fondamentale nella computazione quantica e nella creazione di algoritmi. Alcune di queste porte porte presentano un funzionamento analogo a porte classiche, mentre altre porte, che sfruttano le proprietà quantiche dei qubit, presentano un funzionamento non comparabile ad alcuna porta logica classica.

Pauli Gates

Le porte di Pauli (X, Y, Z) sono tre porte che agiscono su un singolo qubit. Le matrici di Pauli X, Y e Z corrispondono, rispettivamente, a una rotazione attorno agli assi x, y e z della sfera di Bloch di π radianti.

Tra queste tre porte la più importante è la **porta X** in quanto rappresenta l'operazione di NOT, oppure bit-flip. Come la sua controparte classica questa porta è utilizzata in quasi tutti i circuiti quantistici. La matrice che descrive il funzionamento di questa porta è la seguente:

$$X = NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Il funzionamento della porta X può anche essere descritto dalla seguente operazione:

$$a|0\rangle + b|1\rangle \rightarrow b|0\rangle + a|1\rangle.$$

A differenza di altre porte la porta di Pauli non viene rappresentata dalla propria lettera, come accade con altre porte come ad esempio la porta Y e Z, invece viene rappresentata dal seguente simbolo:

Figura 3.1: Rappresentazione della porta NOT nei circuiti reversibili.

La **porta Y** può essere descritta dalla seguente matrice:

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}.$$

E dalla seguente operazione:

$$a|0\rangle + b|1\rangle \rightarrow -ib|0\rangle + ia|1\rangle.$$

Mentre la **porta Z** è una caso particolare della porta **Phase Shift**, porta che ruota il qubit attorno all'asse Z della sfera di Bloch di un angolo ϕ. La porta Z di Pauli può essere rappresentata come la porta Phase Shift con $\phi = \pi$.

$$P(\phi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix},$$

$$P(\pi) = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

E rappresenta la seguente operazione seguente operazione:

$$a|0\rangle + b|1\rangle \rightarrow a|0\rangle - b|1\rangle.$$

Hadamard Gate

La porta Hadamard agisce su un solo qubit. Questa porta non ha una controparte classica, questo perché il suo comportamento sfrutta le proprietà quantiche del qubit mettendolo in una sovrapposizione di stati. Il comportamento può essere descritto come:

$$|0\rangle \rightarrow \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle\,,$$
$$|1\rangle \rightarrow \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle\,.$$

Gli stati che la porta Hadamard restituisce quanto viene applicato allo stato $|0\rangle$ ed $|1\rangle$ vengono chiamati, rispettivamente, $|+\rangle$ e $|-\rangle$. Se si misura un qubit in uno di questi due stati si avrà la medesima probabilità di osservare lo stato $|0\rangle$ o $|1\rangle$, ovvero 50% per ogni stato.

La rappresentazione sotto forma di matrice unitaria della porta Hadamard è la seguente:

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

V Gate

Un altra porta logica quantica che non presenta una controparte classica è la porta V. Come la porta H, anche questa porta applicata ad un qubit porta il suo stato in sovrapposizione. La porta V viene anche chiamata porta $\sqrt{NOT}$, questo perché se viene applicata due volte, avrà lo stesso effetto di aver applicato una volta la porta Pauli-X (ovvero la porta NOT). La sua rappresentazione in forma matriciale è la seguente:

$$V = \frac{1+i}{2} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}.$$

In questo caso non è interessante analizzare lo stato in cui lascia i qubits, invece la cosa importante di questa porta logica sono le sue proprietà che possono essere rappresentate dalle seguenti formule:

$$V * V = V^\dagger * V^\dagger = NOT,$$
$$V * V^\dagger = I.$$

Ovvero, applicando due volte la porta V o $V^\dagger$, lo stato del qubit è equivalente a quello che si avrebbe ottenuto applicando una volta la porta NOT. Inoltre applicare in successione la porta V e la porta $V^\dagger$ al medesimo qubit, non altera il suo stato finale, lasciandolo allo stato che aveva prima dell'applicazione delle due porte.

La porta $V^\dagger$ è rappresentata dalla matrice trasposta coniugata della matrice V, ovvero:

$$V^\dagger = \frac{1-i}{2}\begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}.$$

Control-NOT Gate

A differenza delle porte viste in precedenza, la porta Controllo-NOT agisce su due qubits. Il primo qubit viene chiamato controllo, mentre il secondo viene chiamato bersaglio. La porta C-NOT applica al qubit bersagio la porta NOT se, e solo se, il qubit controllo è nello stato $|1\rangle$. La rappresentazione sotto forma di matrice unitaria della porta C-NOT è la seguente:

$$C - NOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

La porta C-NOT lascia invariato il qubit cd controllo mentre mette il qubit di target in uno stato che rappresenta l'operazione di XOR tra i due input.

La porta C-NOT è anche utilizzata negli algoritmi quantistici per mettere più qubit in uno stato di intreccio quantico(entanglement).

Nello stesso modo in cui il gate C-NOT è la versione con qubit di controllo della porta NOT, è possibile creare la versione *Control-* di tutte le porte quantiche unitarie.

Taffoli Gate

La porta di Taffoli agisce su 3 qubit, questo ha una conseguenza molto importante ossia che non è una porta quantistica elementare, di conseguenza, è costruito utilizzando una combinazione di porte quantistiche elementare.

La porta di Taffoli viene anche chiamato Control-Control-NOT, questo perché il suo funzionamento è simile a quello della porta C-NOT, ma con due qubit di controllo anziché uno. Verrà applicata la porta NOT al qubit target se i due

qubit di input sono entrambi nello stato $|1\rangle$.
Come è facile capire dal suo comportamento, se il qubit di target è nello stato $|0\rangle$, lo stato di questo qubit dopo aver applicato la porta Taffoli sarà quello di tra `AND` i due qubit di controllo.

Nonostante non sia una porta logica elementare si può comunque rappresentare sotto forma di matrice unitaria:

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

La porta di Taffoli ha $QC = 5$, ovvero, per la sua implementazione sono necessarie cinque porte quantiche elementari. Il circuito per la sua implementazione è il seguente:

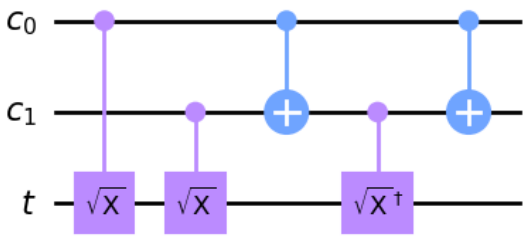


Figura 3.2: Circuito di implementazione della porta di Taffoli.

Dove C_0 e C_1 sono i due qubit di controllo, mentre t è il qubit di target.

Capitolo 4

Square root V Gate

Un circuito di Full-Adder permette di sommare tre qubits allo stesso tempo, questo però può essere migliorato costruendo circuiti che sommino più di tre qubit per volta, per fare ciò però c'è bisogno di introdurre una nuova porta logica, la porta $\sqrt{V}$ Gate.

La porta V svolge l'operazione di $\sqrt{X}$, infatti le sue proprietà ci dicono che appiccandola due volte si svolge l'operazione di NOT. Utilizzando questo concetto, in questo capitolo si propone la porta $\sqrt{V}$, come il nome suggerisce, se applicata due volte si otterrà l'operazione della porta V.

In questo capitolo verrà descritta questa porta, come è stata ricava e le sue proprietà. Nel capitolo successivo verrà mostrato come utilizzare questa porta per creare circuiti che sommino più di tre qubit per volta.

4.1 Rappresentazione

La porta $\sqrt{V}$ è una porta quantistica elementare che agisce su un solo qubit. Similmente ad altre porte viste in precedente il comportamento della porta $\sqrt{V}$ non può essere paragonato a nessuna porta logica classica.

La rappresentazione sotto forma di matrice unitaria della porta $\sqrt{V}$ è la seguente:

$$\sqrt{V} = \frac{2+\sqrt{2}(1+i)}{4} \begin{pmatrix} 1 & i-i\sqrt{2} \\ i-i\sqrt{2} & 1 \end{pmatrix}.$$

4.1.1 Calcolo della matrice

La proprietà fondamentale della porta $\sqrt{V}$ è che se viene applicata due vole svolge la stessa operazione della porta V. A livello matriciale questo si può rappresentare con la seguente uguaglianza:

$$\sqrt{V} * \sqrt{V} = V.$$

E, per le proprietà della porta V, si ha che:

$$\sqrt[4]{X} = \sqrt{V}.$$

Possiamo esprimere la porta NOT partendo da una formula simile a quella di Euler:

$$e^{-i\theta X} = \cos\theta X - \sin\theta X,$$

sostituendo $\theta = \frac{\pi}{2}$ otteniamo:

$$X = ie^{-i\frac{\pi}{2}X} = e^{i\frac{\pi}{2}}e^{-i\frac{\pi}{2}X}.$$

A questo punto possiamo applica radice quarta per ottenere la rappresentazione matriciale della porta $\sqrt{V}$:

$$\begin{aligned}
\sqrt[4]{X} &= \sqrt[4]{e^{i\frac{\pi}{2}} * e^{-i\frac{\pi}{2}X}} \\
&= e^{i\frac{\pi}{8}} * e^{-i\frac{\pi}{8}X} \\
&= e^{i\frac{\pi}{8}} * \left(\cos\frac{\pi}{8}I - i\sin\frac{\pi}{8}X\right) \\
&= e^{i\frac{\pi}{8}} * \left(\cos\frac{\pi}{8}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} + \sin\frac{\pi}{8}\begin{pmatrix}0 & -i\\ -i & 0\end{pmatrix}\right) \\
&= (\cos\frac{\pi}{8} + i\sin\frac{\pi}{8})\begin{pmatrix}\cos\frac{\pi}{8} & -i\sin\frac{\pi}{8}\\ -i\sin\frac{\pi}{8} & \cos\frac{\pi}{8}\end{pmatrix}.
\end{aligned}$$

A questo punto possiamo sostituire $\cos\frac{\pi}{8} = \frac{\sqrt{2+\sqrt{2}}}{2}$ e $\sin\frac{\pi}{8} = \frac{\sqrt{2-\sqrt{2}}}{2}$. Svolgendo qualche semplificazione si ottiene la rappresentazione presentata all'inizio di questo capitolo:

$$\sqrt{V} = \frac{2+\sqrt{2}(1+i)}{4}\begin{pmatrix}1 & i-i\sqrt{2}\\ i-i\sqrt{2} & 1\end{pmatrix}.$$

Per verificare la correttezza di questa matrice, è sufficiente moltiplicare la matrice per se stessa e si ottiene la rappresentazione matriciale della porta V.

4.1.2 $\sqrt{V}^\dagger$ Gate

La porta $\sqrt{V}^\dagger$ svolge l'operazione opposta della porta $\sqrt{V}$. Ovvero applicare $\sqrt{V}$ e $\sqrt{V}^\dagger$ consecutivamente allo stesso qubit equivale a non svolgere alcuna operazione.

La rappresentazione sotto forma di matrice unitaria della porta $\sqrt{V}^\dagger$ è la seguente:

$$\sqrt{V}^\dagger = \frac{1}{1+i(\sqrt{2}-1)} \begin{pmatrix} 1 & i(\sqrt{2}-1) \\ i(\sqrt{2}-1) & 1 \end{pmatrix}.$$

Questa matrice può essere calcolata a partire della matrice di $\sqrt{V}$, in quanto la matrice $\sqrt{V}^\dagger$ è la matrice trasposta coniugata della matrice $\sqrt{V}$.

4.2 Proprietà

La porta $\sqrt{V}$ è interessante per le sue proprietà e per la maniera in cui interagisce con le altre porte. La prima proprietà deriva direttamente dal modo in cui questa porta è calcolata, ovvero:

$$\sqrt{V} * \sqrt{V} = V.$$

Questa equazione ci dice che applicando due vole la porta $\sqrt{V}$ al medesimo qubit, il risultato sarà equivalente a quello che avremmo ottenuto se fosse stata applicata la porta V.

Una proprietà già detta in precedenza ma degna di essere ripetuta è la seguente:

$$\sqrt{V} * \sqrt{V}^\dagger = I.$$

Questo è vero per la definizione stessa di $\sqrt{V}^\dagger$, essa è la matrice trasposta coniugata della matrice V.

Utilizzando queste proprietà se ne possono ricavare altre che ci permettono di descrivere facilmente il comportamento di queste porte logiche in combinazione con la porta V:

$$V * \sqrt{V}^{\dagger} = \sqrt{V},$$
$$V^{\dagger} * \sqrt{V} = \sqrt{V}^{\dagger}.$$

Per dimostrare la prima basti sostituire $V = \sqrt{V} * \sqrt{V}$, ottenendo $\sqrt{V} * \sqrt{V} * \sqrt{V}^{\dagger} = \sqrt{V}$. Come visto in precedenza $\sqrt{V} * \sqrt{V}^{\dagger} = I$, andandolo a sostituire otteniamo $\sqrt{V} * I = \sqrt{V}$ che è corretto, quindi la proprietà è verificata. Una dimostrazione analoga si può fare per $V^{\dagger} * \sqrt{V} = \sqrt{V}^{\dagger}$, sostituendo $V^{\dagger} = \sqrt{V}^{\dagger} * \sqrt{V}^{\dagger}$.

Capitolo 5

N-qubits Adder

Un circuito capace di sommare tre qubits viene chiamato Full-Adder, esistono diverse implementazioni per un circuito del genere [6] [7] [8]. Se si vogliono sommare più di 3 qubits per volta occorre la necessità di utilizzare varie *Half-Adder* e *Full-Adder*. Utilizzando circuiti specifici per la somma di un certo numero di qubit si possono creare circuiti con parametri (come il CI e QC) ridotti.

In questo capitolo verrà mostrato come utilizzare la porta $\sqrt{V}$ per creare circuiti *N-qubit Adder*, che permettano di sommare N qubits alla volta, con $N \leq 7$.

5.1 Principio e nomenclatura

Prima di poter cominciare ad analizzare i circuiti è necessario stabilire una nomenclatura e chiarire il concetto di ordine di grandezza contestualizzato a queste porte.

5.1.1 Ordine di Grandezza

Un Full-Adder restituisce in uscita 2 qubit interessanti: *somma* e *resto*. Il qubit di *somma* rappresenta l'operazione di XOR fra i tre qubits da sommare. Mentre il *resto* rappresenta il qubits di un ordine di grandezza superiore. Ad esempio se si sommano i bit '1', '0' e '1'; l'uscita sarà '10' il bit di somma sarà '0' mentre il bit di resto sarà '1'.

Visualizzando l'uscita come numero unico si ha '10', questo permette di analizzare i singoli valori di uscita in base al loro ordine di grandezza (OoM) rispetto ai bit d'entrata. In questo caso '0' ha lo stesso ordine di grandezza dei tre bit in entrata, mentre '1' è di un ordine di grandezza superiore.

Simbolo	Significato	OoM
U	Unità	2^0
D	Decine	2^1
H	Centinaia	2^2

Tabella 5.1: Ordini di grandezza dei delle cifre di un numero rappresentato da più qubit.

Chiameremo l'ordine di grandezza dei qubit in entrata *U*, assegneremo poi l'ordine di grandezza ai qubit in uscita relativamente a quelli di entrata. Il qubit di somma, che rappresenta sempre l'operazione di XOR tra i qubit *U* in entrata, è dello stesso ordine di grandezza, ovvero *U*. Il qubit di resto, come detto in precedenza, è di un ordine di grandezza superiore, quindi *D*. Se si considera un Full-Adder non c'è alcuna ragione per andare avanti con la nomenclatura, ma in questo capitolo si vedranno circuiti che sommano fino a sette qubit, producendo in output un ulteriore resto dell'ordine di grandezza delle *H*.

Analizzare l'ordine di grandezza non serve solo per identificare i qubit in uscita, nella sezione *XuYd-Adder* (5.3) verranno analizzati adder capaci di sommare qubit di diversi ordini di grandezza.

5.1.2 Nome delle porte

I circuiti utilizzati per sommare due e tre qubit anno nomi noti, Half-Adder e Full-Adder.

Per riferirsi ai circuiti che permettono di sommare più di tre qubit e che utilizzano la porta $\sqrt{V}$ verrà utilizzato il nome 'Xu-Adder', con X il numero di qubit di OoM *U* che permette di sommare il circuito. Quindi, ad esempio, il circuito che permette di sommare cinque qubit si chiama 5u-Adder, mentre quello per sette è 7u-Adder. Questi circuiti verranno analizzati nella sezione *Xu-Adder* (5.2).

Come anticipato, la porta $\sqrt{V}$ permette di costruire circuiti di somma che accettino in input qubit non solo di ordine di grandezza *U*, ma anche *D*, che quindi gestiranno in maniera differente questi qubit in base al loro ruolo. Per identificare con precisione questi circuti e il loro compito si è adoperata la seguente nomenclatura 'XuYd-Adder', con X il numero di qubit di OoM *U* e Y il numero di qubit di OoM *D* che permette di sommare il circuito.

5.2 Xu-Adder

Gli 'Xu-Adder' sono circuiti che consentono di sommare X qubit insieme. La X presenta alcune limitazioni, date dalla seguente disequazione:

$$4 \leq X \leq 7.$$

Il limite superiore $X \leq 7$ è dato dal fatto che se si volesse costruire un circuito 8u-Adder ci sarebbe bisogno di introdurre una porta logica aggiuntiva e il circuito avrebbe bisogno di un CI maggiore. Quindi, ai vini di questo documento, si considereranno circuito di comma per cui $X \leq 7$.

Il limite inferiore di $4 \leq X$ in realtà è pratico, in quanto se si volesse realizzare un 3u-Adder utilizzando la porta $\sqrt{V}$ sarebbe possibile, ma questo circuito si potrebbe considerare come una versione peggiore del già noto Full-Adder.

Questi circuiti permettono di sommare qubits che rappresentano cifre dello stesso ordine di grandezza, come in un normale Full-Adder.

5.2.1 Inputs e Outputs

Un Xu-Adder, con $4 \leq X \leq 7$, avrà sempre $X + 2$ input e output. Gli input sono gli X qubit da sommare e due qubit addizionali che rappresentano il CI del circuito, questo parametro verrà approfondito meglio nella sezione *Parametri* (5.2.4). Come per tutti i circuiti quantistici reversibili il numero di input è uguale al numero di output.

Il circuito ha come output $X - 1$ GO e tre qubit che contengono il risultato dell'operazione di somma. I qubit sono i seguenti:

- Somma : Questo qubit è quello che rappresenta l'operazione di `XOR` fra gli input, l'orine di grandezza di questo qubit è il medesimo degli input, ovvero U;
- Primo resto : Questo qubit è di un ordine di grandezza superiore rispetto a la somma, ovvero D. Questo qubit è comparabile al qubit di resto nei circuiti Full-Adder;
- Secondo resto : Questo qubit è di due ordini di grandezza superiore rispetto a la somma, ossia H.

Mentre gli altri $X - 1$ qubit di output rappresentano i primi X-1 qubit che sono stati inseriti come input. In molte applicazioni di questi circuiti il valore

dei qubit originali non è più di interesse, per questo quei qubit di output possono essere considerati *Garbage Output*.

5.2.2 Circuito di implementazione

Per analizzare l'implementazione del circuito si prende come esempio il circuito *6u-Adder*, che presenta la seguente implementazione:

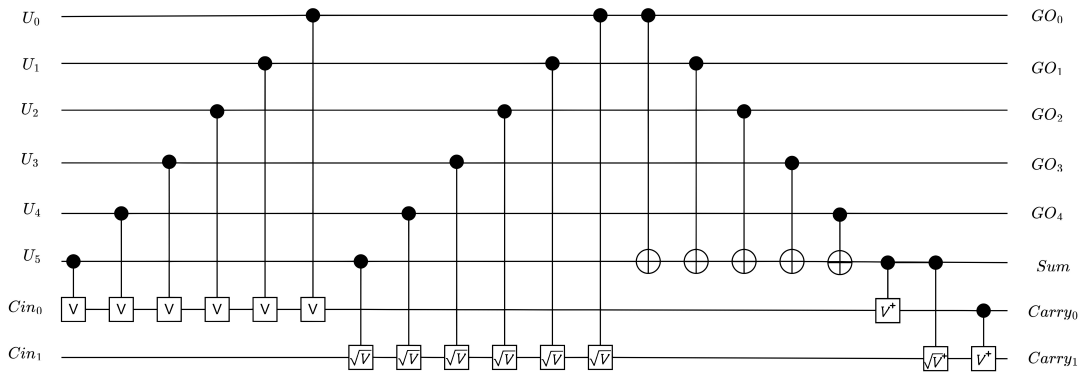


Figura 5.1: Circuito di implementazione di 6u-Adder

Dove gli input sono:

- $U_{0,1,2,3,4,5}$: questi sono i sei qubit in input che verranno sommati;
- $Cin_{0,1}$: questi sono due qubit di CI, necessari per mantenere la reversibilità del circuito. Questi due qubits sono entrambi nello stato $|0\rangle$.

Invece gli output sono:

- $GO_{0,1,2,3,4}$: come si può vedere dal circuito questi qubits non subiscono alterazioni al proprio stato, infatti $U_{0,1,2,3,4} = GO_{0,1,2,3,4}$. Nonostante ciò, questi qubit non forniscono informazioni sull'operazione di addizione, quindi sono da considerare *Garbage Output* (allo stesso modo dei qubit di CI in ingresso il loro scopo è mantenere al reversibilità del circuito);
- Sum: questo qubit rappresenta l'operazione di somma, ovvero l'operazione di `XOR` dei qubit di input U;
- $Carry_0$ questo qubit rappresenta l'operazione di primo resto, con ordine di grandezza D;
- $Carry_1$ questo qubit rappresenta l'operazione di secondo resto, con ordine di grandezza H.

Nel caso degli altri *Xu-Adder*, gli unici parametri che variano sono il numero di qubti U e GO, mentre il numero di qubit Cin, Sum e $Carry$ è lo stesso.

5.2.3 Analisi e funzionamento del circuito

Per analizzare il funzionamento viene diviso in 5 parti come segue:

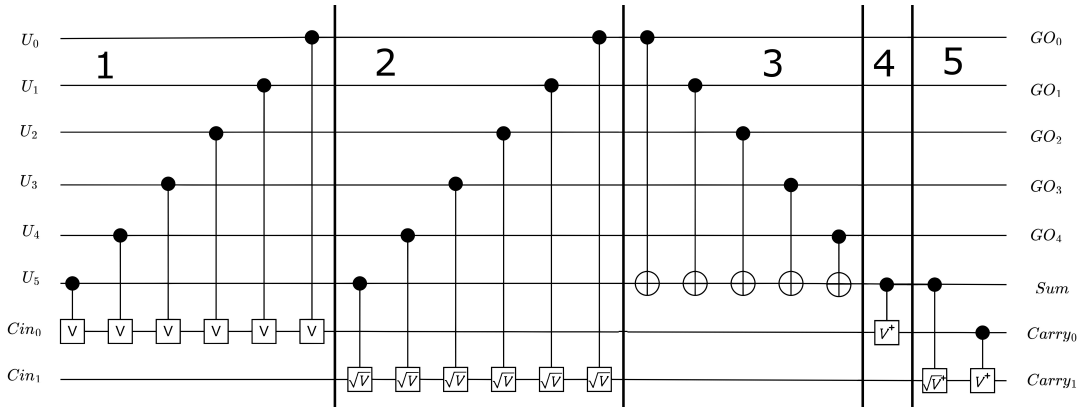


Figura 5.2: Circuito di implementazione di 6u-Adder, diviso per analizzarne il funzionamento.

La parte numero **3** serve per calcolare il qubit di *Sum* e svolge l'operazione più XOR tra i qubits di input da sommare. Tale operazione viene svolta utilizzando le porte C-NOT.

Nella parte numero **1** i qubit da sommare applicano la porta C-V sul qubit Cin_0. Insieme alla porta C-V† applicata nella parte **4**, consente di calcolare il $Carry_0$.

Similmente, nella la parte numero **2** i qubit da sommare applicano la porta C-$\sqrt{V}$ sul qubit Cin_1. Grazie poi alle ultime due porte applicate nella parte **5**, ovvero un C-$\sqrt{V}^\dagger$ e C-V†, viene calcolato il $Carry_1$.

5.2.4 Parametri

Quanto riguarda i parametri delle porte Xu-Adder, con $4 \leq X \leq 7$, si possono determinare semplicemente sapendo il numero X di qubit da sommare:

- **CI = 2** : Il numero di qubit addizionali necessari per l'utilizzo di queste porte è due, indipendentemente da dal numero X di qubit da sommare. Questi due qubits devono essere entrambi nello stato $|0\rangle$.
- **GO = X - 1**: Il numero di qubit di Go dipende dal numero di qubit che vengono sommati in un circuito.

- **QC = 3X + 2**: Il QC, può essere calcolato utilizzando la formula. Si noti che l-incremento del costo è lineare rispetto al numero di input.

Utilizzando queste formule si possono determinare i parametri ti tutte le porte Xu-Adder (con $4 \leq X \leq 7$):

Porta	X	CI	GO	QC
4u-Adder	4	2	3	14
5u-Adder	5	2	4	17
6u-Adder	6	2	5	20
7u-Adder	7	2	6	23

Tabella 5.2: Parametri delle porte Xu-Adder, con $4 \leq X \leq 7$.

5.3 XuYd-Adder

Gli 'XuYd-Adder' sono circuiti che consentono di sommare X qubit del medesimo ordine di grandezza, ovvero U, e Y qubit di un ordine di grandezza superiore rispetto agli X qubit, ovvero D.

Come per gli Xu-Adder, anche gli XuYd-Adder presentano una limitazione nella scelta dei suoi parametri, questa limitazione è data dalla seguente disequazione:

$$X + 2Y \leq 7.$$

Questo vuol dire che si possono costruire circuiti come '4u1d-Adder' e '3u2d-Adder', ma non '6u1d-Adder' e '3u3d-Adder'. La ragione per cui è presente questo limite superiore è la medesima del limite superiore presente nella costruzione dei circuiti Xu-Adder (5.2). Non è presente un vero e proprio limite inferiore per i parametri, ma si può fare un ragionamento analogo a quello dei circuiti Xu-Adder. Si considerino i seguenti casi speciali:

- Se $Y = 0$ allora il circuito sarà un normale Xu-Adder;

- se $X = 0$ allora il circuito può essere sostituito da un Xu-Adder applicato ai qubit di ordine D;

- se $X = 1$ allora l'output di *Sum* sarà uguale al qubit di OoM U e non influirà sul calcolo di $Carry_0$ e $Carry_1$, quindi può essere omesso. Così facendo si torna al caso di $X = 0$.

Si nota che, in accordo con queste osservazioni, Y potrà assumere solo valori di 1 o 2.

5.3.1 Inputs e Outputs

Gli input e output dei circuiti XuYd Adder sono molto simili a quelli dei circuiti Xu-Adder (5.2.1).Un XuYd Adder, avrà sempre $X + Y + 2$ input e output.

Gli input presentano una chiara differenza, ovvero un tipo aggiuntivo di input. Gli input sono:

- X qubit dell'ordine di grandezza U;
- Y qubit dell'ordine di grandezza D;
- 2 CI nello stato $|0\rangle$, questi qubit servono a preservare la reversibilità del circuito.

Il circuito ha come output $X + Y - 1$ GO e, come per i circuiti Xu-Adder, tre qubit che contengono il risultato dell'operazione di somma: *Sum*, $Carry_0$ e $Carry_1$.

5.3.2 Circuito di implementazione

Per analizzare l'implementazione del circuito si prende come esempio il circuito 4u1d-Adder, che presenta la seguente implementazione:

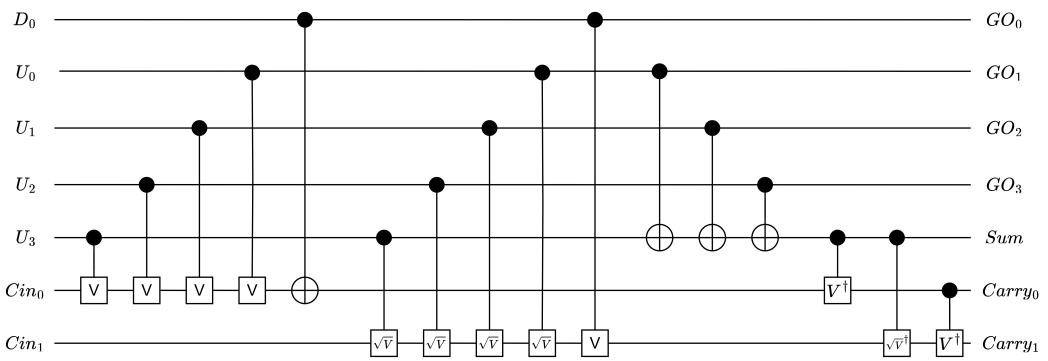


Figura 5.3: Circuito di implementazione di 4u1d-Adder.

Dove gli input sono:

- D_0 : questo è il qubit da sommare con ordine di grandezza D;
- $U_{0,1,2,3}$: questi sono i qubit da sommare con ordine di grandezza U;
- $Cin_{0,1}$: questi sono due qubit di CI, necessari per mantenere la reversibilità del circuito. Questi due qubits sono entrambi nello stato $|0\rangle$.

Invece gli output sono:

- $GO_{0,1,2,3}$: come si può vedere dal circuito questi qubits non subiscono alterazioni al proprio stato, infatti $D_0 = GO_0$ e $U_{0,1,2} = GO_{1,2,3}$. Nonostante ciò, questi qubit non forniscono informazioni sull'operazione di addizione, quindi sono da considerare *Garbage Output*;
- Sum: questo qubit rappresenta l'operazione di somma;
- $Carry_0$ questo qubit rappresenta l'operazione di primo resto;
- $Carry_1$ questo qubit rappresenta l'operazione di secondo resto.

Nel caso degli altri *XuYd-Adder*, gli unici parametri che variano sono il numero di qubti U, D e GO, mentre il numero di qubit Cin, Sum e $Carry$ è lo stesso.

5.3.3 Analisi e funzionamento del circuito

L'analisi dei circuiti *XuYd-Adder* è analoga all'analisi dei circuiti *Xu-Adder* svolta nel sezione *Analisi e funzionamento del circuito* (5.2.3).

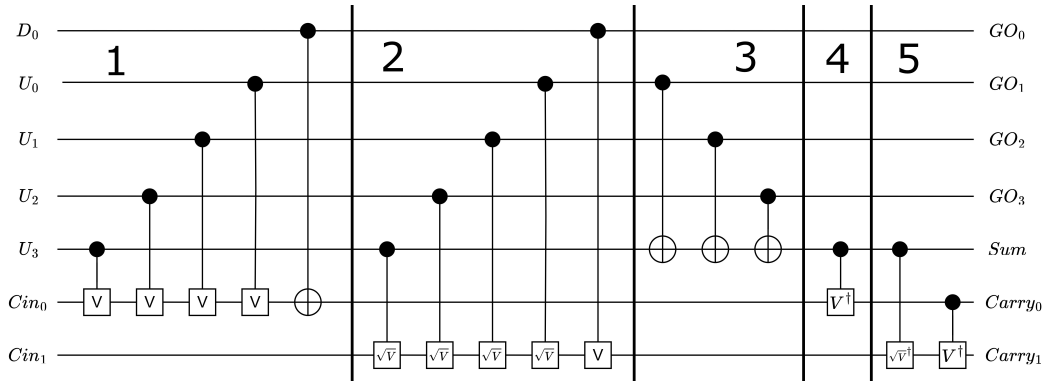


Figura 5.4: Circuito di implementazione di 4u1d-Adder, diviso per analizzarne il funzionamento.

Ai qubit $U_{0,1,2,3}$ vengono applicate le stesso porte e hanno lo stesso funzionamento che hanno in un circuito Xu-Adder. La differenza in questo circuito

è data dai qubit D (nell'esempio ne è presente uno), che svolgono operazioni paragonabili a quelle dei qubit U, ma non uguali.

Nella sezione **1**, invece di applicare una porta $C-V$ al qubit Cin_0, i qubit D applicano una porta $C-NOT$. Nella sezione **2**, invece di applicare una porta $C-\sqrt{V}$ al qubit Cin_1, i qubit D applicano una porta $C-V$. Per ultimo i qubit D non applicano alcuna porta nella sezione **3**. Le ultime due sezioni rimangano invariate.

5.3.4 Parametri

Anche per i circuiti XuYu-Adder sono presenti formule per determinare i parametri a partire dal valore di X e Y:

- **CI = 2** : Il numero di qubit addizionali rimane costante a due, indipendentemente da dal numero X e Y. Questi due qubits devono essere entrambi nello stato $|0\rangle$.
- **GO = X + Y - 1**: Il numero di qubit di Go dipende dal numero di qubit che vengono sommati in un circuito.
- **QC = 3X + 2Y + 2**: Il QC, può essere calcolato utilizzando la formula.

Quindi, ad esempio, il circuiti 4u1d-Adder visto in precedenza ha $CI = 2$, $GO = 4$ e $QC = 16$.

Capitolo 6

Simulazione di XuYd-Adder

Qiskit è una libreria, per il linguaggio Python, che consente di creare e analizzare circuiti quantistici.

Qiskit consente di visualizzare i circuiti creati tramite interfaccia grafica, consentendo di comprendere ed analizzare meglio gli stati dei qubit e il funzionamento delle porte logiche. L'utilizzo di simulatori è utile per verificare il corretto funzionamento dei calcoli sperimentali, in quanto i computer quantistici sono rari e di difficile accesso.

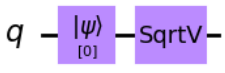
6.1 $\sqrt{V}$

Per poter simulare i circuiti XuYd-Adder prima c'è bisogno di definire la porta $\sqrt{V}$. Questo è possibile tramite la classe `Operator()`.

```
1 sqrtVSup = (2 + np.sqrt(2)*(1+1j))/4
2
3 sqrtV = Operator([[sqrtVSup, sqrtVSup*1j *(1-np.sqrt(2))],
4                   [sqrtVSup*1j*(1-np.sqrt(2)), sqrtVSup]])
```

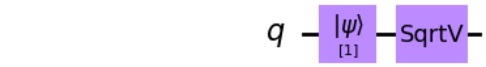
Listing 6.1: Codice epr l'implementazione della porta $\sqrt{V}$.

Una volta implementata possiamo utilizzarla per simulare il suo comportamento quando applicato ad un qubit.



$$(0.8535533906 + 0.3535533906i)|0\rangle + (0.1464466094 - 0.3535533906i)|1\rangle$$

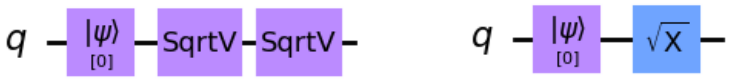
Figura 6.1: Circuito di output della porta $\sqrt{V}$ Gate applicata ad un qubit nello stato $|0\rangle$.



$$(0.1464466094 - 0.3535533906i)|0\rangle + (0.8535533906 + 0.3535533906i)|1\rangle$$

Figura 6.2: Circuito di output della porta $\sqrt{V}$ Gate applicata ad un qubit nello stato $|1\rangle$.

Come detto in precedenza lo stato risultante non è ciò che interessa di questa porta, ma lo sono le sue proprietà. Per testare il corretto funzionamento delle proprietà applicheremo due volte la porta $\sqrt{V}$, ad un circuito nello stato $|0\rangle$, e confronteremo il risultato con un circuito in cui invece viene applicata una volta la porta V. Verifichiamo che le proprietà teoriche sono corrette in quanto lo stato di output del qubit è il medesimo.



$$(\frac{1}{2} + \frac{i}{2})|0\rangle + (\frac{1}{2} - \frac{i}{2})|1\rangle$$

Figura 6.3: Circuito della porta $\sqrt{V}$ applicata due volte e della porta V, con il loro output.

Con questo è stata simulata la porta $\sqrt{V}$ e ne sono state verificate le proprietà, che verranno poi sfruttate per la creazione di circuiti.

6.2 XuYd-Adder

Con l'utilizzo della porta $\sqrt{V}$ si possono simulare le porte XuYd-Adder. L'obbiettivo è verificare il corretto funzionamento della porta. Tramite qiskit non solo si possono visualizzare in forma grafica i circuiti implementati, ma si possono anche simulare multipli stati. Questo consente di simulare tutti i possibili stati classici di input.

Si noti che è possibile simulare anche stati di sovrapposizione, essendo infiniti non è possibile simularli tutti. Siccome il circuito svolge un'operazione classica, la somma, è sufficiente verificare la correttezza del circuito nel caso di input con stati classici.

Verrà verificato il comportamento del circuito 4u1d-Adder, che viene implementato col seguente circuito:

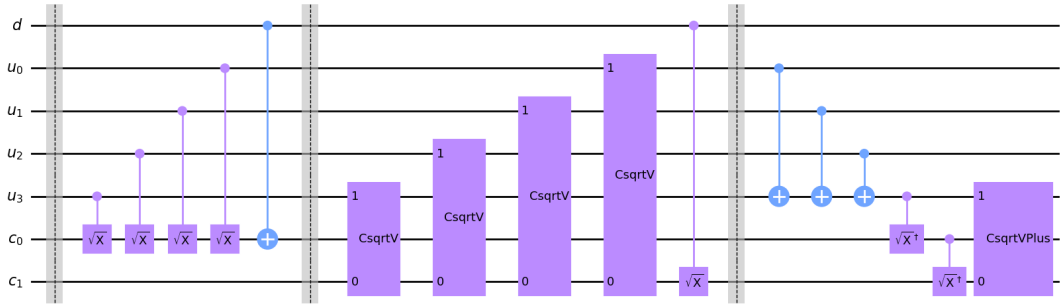


Figura 6.4: Rappresentazione del circuito 4u1d con qiskit.

Per poter verificare il corretto funzionamento ella porta vengono simulati tutti i possibili stati classici dei qubit in input e viene verificato se l'output è quello atteso.

L'input della porta sono sette qubit, lo stato di input può quindi essere rappresentato come $|0000000\rangle$, nel caso di tutti qubit nello stato $|0\rangle$. Non tutti i qubit vengono trattati allo stesso modo, l'ordine dei qubit in input è il seguente $|C_1C_0U_3U_2U_1U_0D\rangle$, dove:

- C è il *CI*, che deve essere impostato a $|00\rangle$;
- U sono i qubit con ordine di grandezza U;
- D è il qubit con ordine di grandezza D;

Quindi, ad esempio, l'input per sommare due qubit con ordine di grandezza U e un qubit con ordine di grandezza D sarà $|0011001\rangle$ oppure $|0010011\rangle$.

Allo stesso modo dell'input, anche l'output della porta sono sette qubit e possiamo interpretare l'output come $|C_1C_0SG_3G_2G_1G_0\rangle$, dove:

- C_1 è il qubit che rappresenta il $Carry_1$;
- C_0 è il qubit che rappresenta il $Carry_0$;
- S è il qubit che rappresenta la Sum;
- G sono i qubit di *GO*.

Non è necessario verificare tutti i 2^7 possibili input classici, è sufficiente verificare i 2^5 input nei quali i primi due qubit sono nello stato $|00\rangle$. Questo perché questi qubit sono di CI quindi è necessario che abbiano un valore di input prestabilito in tutte le esecuzioni del circuito. Si potrebbe eseguire il circuito con CI differente, ma non svolgerebbe più la funzione di sommatore.

Con l'utilizzo di qiskit, è possibile eseguire il circuito 2^5 volte cambiando il valore dei qubit in ingresso, per simulare tutti gli input desiderati.

A causa dell'approssimazione dei numeri reali con numeri di macchina e degli errori di stabilità delle operazioni, la simulazione del circuito presenta qualche imprecisione, come mostrato negli esempi seguenti.

```
Initial state: 0000111
{'0000111': 1.9259299443872359e-32, '1000111': 1.0}
```

Figura 6.5: Output del circuito 4u1d-Adder con input $|0000111\rangle$ ed errore di macchina.

```
Initial state: 0011111
{'0101111': 1.3595030923912448e-31, '1101111': 1.0}
```

Figura 6.6: Output del circuito 4u1d-Adder con input $|0011111\rangle$ ed errore di macchina.

Qiskit restituisce in output con probabilità "1.0" la risposta corretta, ma restituisce anche in output con probabilità dell'ordine di 10^{-32} una risposta errata. Questo non è un errore causato del blocco 4u1d o dalla sua implementazione. Questo errore è invece causato dall'approssimazione dei numeri reali e immaginari della macchina utilizzata. Questa approssimazione può causare errori quando si svolge l'operazione di sottrazione, operazione instabile che può causare errori arbitrariamente grandi su numeri approssimati.

I risultai ottenuti sono quindi stati puliti dai questi errori di stabilità e si considera solo l'output con probabilità "1.0". Mettendo in relazione gli input con i corrispettivi output ottenuti si ottiene:

$$
\begin{aligned}
|0000000\rangle &\to |0000000\rangle & \qquad |0010000\rangle &\to |0010000\rangle \\
|0000001\rangle &\to |0100001\rangle & \qquad |0010001\rangle &\to |0110001\rangle \\
|0000010\rangle &\to |0010010\rangle & \qquad |0010010\rangle &\to |0100010\rangle \\
|0000011\rangle &\to |0110011\rangle & \qquad |0010011\rangle &\to |1000011\rangle \\
|0000100\rangle &\to |0010100\rangle & \qquad |0010100\rangle &\to |0100100\rangle \\
|0000101\rangle &\to |0110101\rangle & \qquad |0010101\rangle &\to |1000101\rangle \\
|0000110\rangle &\to |0100110\rangle & \qquad |0010110\rangle &\to |0110110\rangle \\
|0000111\rangle &\to |1000111\rangle & \qquad |0010111\rangle &\to |1010111\rangle
\end{aligned}
$$

$$|0001000\rangle \rightarrow |0011000\rangle \qquad |0011000\rangle \rightarrow |0101000\rangle$$
$$|0001001\rangle \rightarrow |0111001\rangle \qquad |0011001\rangle \rightarrow |1001001\rangle$$
$$|0001010\rangle \rightarrow |0101010\rangle \qquad |0011010\rangle \rightarrow |0111010\rangle$$
$$|0001011\rangle \rightarrow |1001011\rangle \qquad |0011011\rangle \rightarrow |1011011\rangle$$
$$|0001100\rangle \rightarrow |0101100\rangle \qquad |0011100\rangle \rightarrow |0111100\rangle$$
$$|0001101\rangle \rightarrow |1001101\rangle \qquad |0011101\rangle \rightarrow |1011101\rangle$$
$$|0001110\rangle \rightarrow |0111110\rangle \qquad |0011110\rangle \rightarrow |1001110\rangle$$
$$|0001111\rangle \rightarrow |1011111\rangle \qquad |0011111\rangle \rightarrow |1101111\rangle$$

Questi valori si possono utilizzare ed analizzare come se fossero una tabella di verità, ricordando che non rappresenta tutti i possibili input e output del circuito.

In particolare i quattro qubit di GO rappresentano sempre gli ultimi quattro qubit di input, come visto questi qubit non presentano il loro stato alterato da porte quantiche.

Mentre i primi tre qubit di output corrispondono alle operazioni di $Carry_1$, $Carry_0$ e Sum; come da previsione.

A seguito delle simulazioni si conclude che il circuito 4u1d-Adder presenta il comportamento aspettato, permettendo di sommare qubit con diversi ordini di grandezza.

Capitolo 7

Implementazione di XuYd-Adder in un circuito di moltiplicazione

I circuiti di moltiplicazione si possono dividere in due parti, la parte di moltiplicazione bit a bit e la parte in cui si sommano tutti i sub prodotti. Prendendo come esempio il prodotto di due numeri composti da quattro qubit ciascuno, X composto dai qubit x_3, x_2, x_1, x_0 e Y composto dai qubit $= y_3, y_2, y_1, y_0$.

La parte di moltiplicazione svolge l'operazione di AND tra i qubit e da come output 16 qubits di diverso ordine di grandezza da dover sommare per poter ottenere il risultato.

				x_3	x_2	x_1	x_0
			x	y_3	y_2	y_1	y_0
				x_3y_0	x_2y_0	x_1y_0	x_0y_0
			x_3y_1	x_2y_1	x_1y_1	x_0y_1	
		x_3y_2	x_2y_2	x_1y_2	x_0y_2		
	x_3y_3	x_2y_3	x_1y_3	x_0y_3			
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

Figura 7.1: Rappresentazione dei calcoli da svolgere per sommare due numeri composti da quattro qubit.

Per la parte di moltiplicazione si possono usare porte *Peres Gate* [9] per svolgere l'operazione di AND tra qubit. Questa operazione utilizza 16 porte *Peres Gate*.

Mentre per la parte di somma dei qubit si possono usare circuiti costruiti utilizzando la porta $\sqrt{V}$, in particolare si utilizza un *4u1d-Adder* ed un *6u-Adder*. Nel circuito vengono utilizzati anche le porte *Peres Gate* e *Peres Fulll-Adder Gate*. La seguente è la configurazione di porte utilizzata per l'operazione di somma.

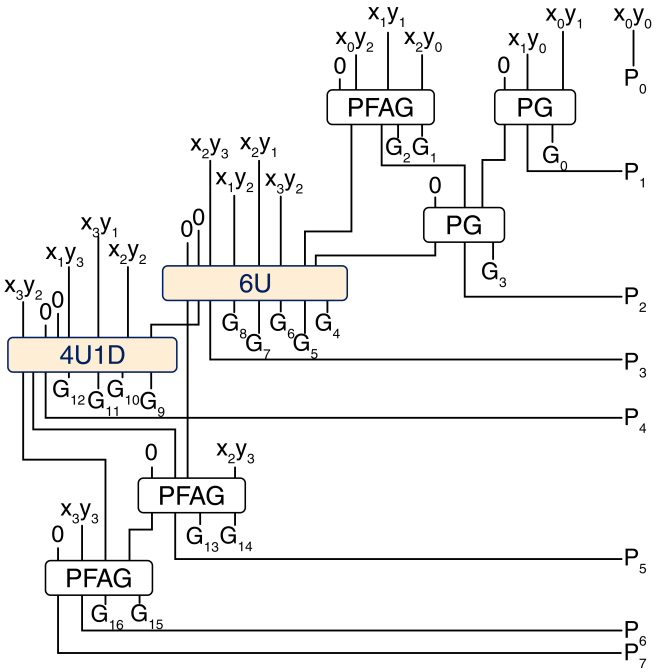


Figura 7.2: Rappresentazione del circuito di somma.

7.1 Analisi e comparazione dei parametri

Si possono calcolare i parametri dei due circuiti sommando i parametri delle porte che compongoino i circuiti.
Moltiplicazione:

- $QC = 16PG = 16 \times 4 = 64$;
- $CI = 16PG = 16$;
- $GO = 8$.

Somma:

- $QC = 2PG + 3PFAG + 6U + 4U1D = 2 \times 4 + 3 \times 6 + 20 + 16 = 62$;
- $CI = 2PG + 3PFAG + 6U + 4U1D = 2 + 3 + 2 + 2 = 9$;

- $GO = 2PG + 3PFAG + 6U + 4U1D = 2 + 2 \times 3 + 5 + 4 = 17$.

Sommando i parametri delle due parti si ottiene:

$$QC = 64 + 62 = \mathbf{126},$$

$$CI = 16 + 9 = \mathbf{25},$$

$$GO = 8 + 17 = \mathbf{25}.$$

Confrontando questi parametri con altri circuiti di moltiplicazione, notiamo un lieve miglioramento del QC e un sostanziale miglioramento nel CI. Questo miglioramento è dato dalla parte di addizione in quanto la parte di moltiplicazione è la medesima di [9]. Considerando solo la parte di somma il circuito di Riferimento [9] presenta $CI = 12$ e $QC = 64$, mentre quello proposto $CI = 9$, con un miglioramento del 25%, e $QC = 62$, con un miglioramento di circa il 3%. Confrontando questo circuito nella sua totalità con altre proposte, si ricavano i seguenti dati:

Circuito	QC	GO	CI
Circuito proposto	126	25	25
Riferimento [9]	128	28	28
Riferimento [10]	160	28	44
Riferimento [11]	256	52	48

Tabella 7.1: Comparazione dei parametri del circuito proposto con i riferimenti.

Questo esempio serve per dimostrare l'utilità dei circuiti XuYd-Adder. La riduzione del CI è possibile grazie ai circuiti 4u1d-Adder e 6u-Adder, che permettono di fare somme, che richiederebbero tre o quattro circuiti di Half-Adder o Full-Adder, con un $CI = 2$.

I circuiti XuYd-Adder sono un esempio delle applicazioni della porta $\sqrt{V}$, con ulteriore ricerca potrebbe rilevarsi utile anche in circuiti al di fuori di quelli di somma.

Capitolo 8

Conclusioni

La computazione quantica è ancora agli inizi del suo sviluppo, l'implementazione dei qubit è limitata e presenta numerose sfide. Per implementare circuiti sempre migliori non c'è solo bisogno di un avanzamento nella realizzazione di questi calcolatori, sono anche necessari circuiti ottimizzati che richiedano il minor numero possibile di qubit e porte logiche.

È quindi importante ottimizzare i circuiti reversibili, riducendo i suoi parametri. In questa tesi si è proposto l'utilizzo della porta $\sqrt{V}$ per creare nuovi circuiti con parametri ridotti. Come si è visto, questa porta permette di ridurre il numero di qubit necessari per l'implementazione dei circuiti di somma, rendendoli più facili da implementare.

Nella logia reversibile si hanno a disposizione un infinito numero di porte che permettono di sfruttare le proprietà del qubit. Questo conferisce alla computazione quantica un grosso vantaggio rispetto alla computazione classica. È quindi importante che queste proprietà vengano utilizzate nella realizzazione di circuiti reversibili, evitando di limitarsi a tradurre i circuiti classici.

Bibliografia

[1] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM journal of research and development*, vol. 5, no. 3, pp. 183–191, 1961.

[2] C. H. Bennett, "Logical reversibility of computation," *IBM journal of Research and Development*, vol. 17, no. 6, pp. 525–532, 1973.

[3] M. A. Nielsen and I. L. Chuang, "Quantum computation and quantum information," *Phys. Today*, vol. 54, no. 2, p. 60, 2001.

[4] M. Ehsanpour, P. Moallem, and A. Vafaei, "Design of a novel reversible multiplier circuit using modified full adder," in *2010 International Conference On Computer Design And Applications*, vol. 3. IEEE, 2010, pp. V3–230.

[5] T. Toffoli, "Reversible computing," in *International colloquium on automata, languages, and programming*. Springer, 1980, pp. 632–644.

[6] A. Peres, "Reversible logic and quantum computers," *Physical review A*, vol. 32, no. 6, p. 3266, 1985.

[7] M. Haghparast and K. Navi, "A novel reversible full adder circuit for nanotechnology based systems," *Journal of Applied Sciences*, vol. 7, no. 24, pp. 3995–4000, 2007.

[8] M. S. Islam, M. M. Rahman, Z. Begum, and M. Z. Hafiz, "Low cost quantum realization of reversible multiplier circuit," *Information technology journal*, vol. 8, no. 2, pp. 208–213, 2009.

[9] M. Ehsanpour and P. Moallem, "A novel design of reversible multiplier circuit," *International Journal of Engineering*, vol. 26, no. 6, pp. 577–586, 2013.

[10] M. Haghparast, M. Mohammadi, K. Navi, and M. Eshghi, "Optimized reversible multiplier circuit," *Journal of Circuits, Systems, and Computers*, vol. 18, no. 02, pp. 311–323, 2009.

[11] M. Haghparast, S. J. Jassbi, K. Navi, and O. Hashemipour, "Design of a novel reversible multiplier circuit using hng gate in nanotechnology," in *World Appl. Sci. J.* Citeseer, 2008.