POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Elettronica

Tesi di Laurea Magistrale

Scomposizione di matrici unitarie per la realizzazione di circuiti quantistici in OpenQASM 2.0

Relatori:
Prof. Maurizio Zamboni
Prof. Mariagrazia Graziano
Prof. Giovanna Turvani

Candidato:
Luca Nurisso

Dicembre 2021

Sommario

L'OpenQASM 2.0 è un linguaggio per la descrizione di circuiti quantistici che si distingue per portabilità, in quanto adottato da più framework per la progettazione e l'esecuzione di algoritmi su hardware reale o simulatori. Uno dei limiti di questo linguaggio è la mancanza di supporto per una descrizione comportamentale basata su matrici unitarie.
All'interno del laboratorio VLSI del Politecnico di Torino è in fase di sviluppo una libreria Python per la generazione di circuiti descritti in OpenQASM 2.0 adoperabili in un contesto di progettazione modulare. La libreria è indipendente da qualsiasi tipo di piattaforma proprietaria, garantendone così la portabilità. L'obiettivo principale della tesi è l'integrazione nella libreria di un'infrastruttura automatica per la sintesi di circuiti quantistici, partendo da una loro descrizione mediante matrici unitarie. In particolare, data in ingresso all'infrastruttura una matrice unitaria, attraverso l'esecuzione di diversi step di scomposizione e traduzione, si ottiene in uscita il circuito quantistico corrispondente descritto in OpenQASM 2.0. L'algoritmo di scomposizione sviluppato può essere suddiviso in tre passi principali.
Il primo consiste nell'applicare il metodo di scomposizione delle matrici unitarie proposto da Chi Kwong Li e Diane Christine Pelejo, "International Journal of Quantum Information", vol. 12, no. 01 (2014), ottenendo in uscita un circuito equivalente costituito esclusivamente da porte a singolo qubit o multi-controllo singolo-target. Poiché in OpenQASM 2.0 sono supportate solo porte controllate di tipo singolo-controllo singolo-target, il secondo passo dell'algoritmo prevede una scomposizione ricorsiva delle porte multi-controllo singolo-target per ottenere il formato delle porte desiderato.
La scomposizione ricorsiva non è da sola sufficiente per descrivere il circuito in OpenQASM 2.0, dal momento che l'evoluzione unitaria del target delle porte controllate non corrisponde alla porta U3 nativa del linguaggio. Pertanto, l'ultimo step si occupa della costruzione OpenQASM 2.0-compatibile di tutte le porte del circuito, operando su un contributo di fase relativa tra gli elementi complessi della matrice di evoluzione unitaria del target. Questo metodo di scomposizione delle matrici può essere utilizzato per semplificare la realizzazione di diversi circuiti quantistici, come quelli per la routine di Quantum Phase Estimation o per inizializzare lo stato dei

qubit all'inizio di un algoritmo quantistico, oppure ancora per generare circuiti partendo da matrici unitarie associate a look-up table.

Nel capitolo 1 è riportata una breve introduzione al quantum computing ed al suo formalismo. Invece, nel capitolo 2 è descritto il linguaggio OpenQASM 2.0 ed è una descrizione generale dell'infrastruttura implementata. L'implementazione dell'algoritmo di Li-Pelejo è presentata nel capitolo 3, riportando anche i risultati, in termini di latenza, numero di porte utilizzate ed errore numerico generato, ottenuti per la scomposizione di matrici casuali generiche. Nel capitolo 4 sono descritte le implementazioni dell'algoritmo di scomposizione ricorsivo e del recupero di fase, esponendo i risultati ottenuti scomponendo matrici casuali generiche. Un'applicazione dell'infrastruttura implementata per la realizzazione di un algoritmo quantistico reale è riportata nel capitolo 5 dove sono analizzate le prestazioni ottenute nella realizzazione di due macro-blocchi necessari per l'algoritmo di Quantum Phase Estimation. Nel capitolo 6 sono riportati i risultati per la scomposizione di matrici associate a delle look-up table, in particolare agli operatori aritmetici lineari complemento a due, modulo e reciproco. Infine, il capitolo 7 riporta le conclusioni generali sull'infrastruttura implementata ed introduce alcune considerazioni sulle possibili evoluzioni future.

Indice

Sommario I

1 Introduzione al Quantum Computing **1**
1.1 Il Qubit . . . 1
1.2 Sistemi con più qubit . . . 3
1.3 Matrici Hermitiane e Unitarie . . . 5
1.4 Introduzione alle principali porte quantistiche . . . 7
1.4.1 Porta di Pauli X . . . 8
1.4.2 Porta CX . . . 10
1.4.3 Porta CCX . . . 11
1.4.4 Porta U1 . . . 14
1.4.5 Porta CU1 . . . 15
1.4.6 Porta U3 . . . 16
1.4.7 Porta CU3 . . . 16

2 OpenQASM 2.0 e descrizione generale dell'infrastruttura sviluppata **18**
2.1 Limiti dell'OpenQASM 2.0 . . . 20
2.2 Soluzione proposta . . . 21

3 Scomposizione di Li-Pelejo **24**
3.1 Teoria . . . 24
3.2 Implementazione . . . 32
3.2.1 Diagramma di flusso principale . . . 32
3.2.2 Calcolo dell'ordine di eliminazione e creazione dello schematico 34
3.2.3 Calcolo delle matrici T_i per l'eliminazione degli elementi fuori dalla diagonale. . . . 36
3.3 Semplificazione dello schematico A e di V . . . 38
3.3.1 Inversione dell'ordine di A e V . . . 40
3.4 Risultati . . . 42
3.4.1 Tempo di scomposizione . . . 42

3.4.2 Errore numerico . . . 45
3.4.3 Tabella riassuntiva . . . 47

4 Scomposizione di Nielsen-Chuang e recupero di fase 48
4.1 Teoria . . . 48
4.1.1 Scomposizione ricorsiva di Nielsen-Chuang . . . 48
4.1.2 Recupero di fase . . . 50
4.2 Implementazione . . . 52
4.2.1 Implementazione dell'algoritmo di Nielsen-Chuang . . . 52
4.2.2 Implementazione dell'algoritmo di generazione dell'OpenQASM 2.0 e recupero di fase . . . 58
4.3 Risultati . . . 64
4.3.1 Tempo di scomposizione . . . 64
4.3.2 Risultati complessivi della scomposizione . . . 77
4.4 Metodo di verifica funzionale dell'algoritmo di scomposizione . . . 87

5 Quantum Phase Estimation 91
5.1 Preparazione di stato . . . 94
5.1.1 Teoria . . . 94
5.1.2 Implementazione . . . 96
5.1.3 Risultati . . . 99
5.2 Rotazioni controllate CU^{2^i} . . . 105
5.2.1 Implementazione . . . 105
5.2.2 Risultati . . . 108
5.3 Metodo di verifica funzionale dell'implementazione della QPE . . . 113

6 Evoluzioni unitarie associate a look-up table 118
6.1 Operatore di complemento a due . . . 119
6.1.1 Implementazione . . . 120
6.1.2 Risultati . . . 122
6.2 Operatore di modulo . . . 127
6.2.1 Implementazione . . . 129
6.2.2 Risultati . . . 129
6.3 Operatore di reciproco . . . 136
6.3.1 Implementazione . . . 138
6.3.2 Risultati . . . 140

7 Conclusioni e prospettive future 144

Elenco delle tabelle

3.1 Tabella riassuntiva con i valore delle scomposizione di matrici unitarie tramite l'algoritmo di Li-Pelejo. 47
4.1 Tabella riassuntiva con tutti i risultati delle latenze di scomposizione dei diversi algoritmi adoperati. 85
4.2 Tabella riassuntiva con tutti i risultati delle latenze totale di scomposizione. 86
4.3 Tabella riassuntiva con tutti i risultati degli errori numerici. 86
4.4 Tabella riassuntiva con tutti i risultati del numero di porte adoperate per la descrizione del circuito in OpenQASM 2.0. 86

Elenco delle figure

1.1 Sfera di Bloch. 3
1.2 Rotazione sulla sfera di Bloch della porta X. 10
1.3 Rotazione sulla sfera di Bloch della porta $U1$. 14
2.1 Schema a blocchi raffigurante i tre passi di scomposizione dell'algoritmo implementato. 22
3.1 Diagramma di flusso per l'implementazione dell'algoritmo di scomposizione di Li-Pelejo. 33
3.2 Esempio generico di A e V in uscita dall'algoritmo di scomposizione di Li-Pelejo implementato. 34
3.3 Diagramma di flusso del blocco "calcolo dell'ordine di scomposizione e creazione dello schematico" dell'algoritmo di Li-Pelejo implementato. 35
3.4 Esempio di y, x e A in uscita dal primo blocco di scomposizione dell'algoritmo di Li-Pelejo. 36
3.5 Diagramma di flusso per l'implementazione del blocco di "Calcolo delle matrici T_i per l'eliminazione degli elementi fuori dalla diagonale" dell'algoritmo di scomposizione di Li-Pelejo. 37
3.6 Diagramma di flusso per l'implementazione del blocco di "Semplificazione dello schematico A e della lista di liste V" dell'algoritmo di scomposizione di Li-Pelejo. 39
3.7 Esempio di liste A e V prima dell'operazione di inversione. 40
3.8 Esempio di liste A e V dopo dell'operazione di inversione. 41
3.9 Diagramma di flusso per l'implementazione del blocco di "inversione dell'ordine di a e V" dell'algoritmo di scomposizione di Li-Pelejo. . . 41
3.10 Tempo di scomposizione secondo l'algoritmo di Li-Pelejo di matrici unitarie casuali, in scala lineare. 43
3.11 Tempo di scomposizione secondo l'algoritmo di Li-Pelejo di matrici unitarie casuali, in scala semi-logaritmica. 43
3.12 Interpolazione del tempo di scomposizione secondo l'algoritmo di Li-Pelejo di matrici unitarie casuali, in scala lineare. 44
3.13 Interpolazione del tempo di scomposizione secondo l'algoritmo di Li-Pelejo di matrici unitarie casuali, in scala semi-logaritmica. 44

3.14 Errore numerico riscontrato nelle simulazioni dell'algoritmo di scomposizione proposto da Li-Pelejo. 46
3.15 Interpolazione dell'errore numerico riscontrato nelle simulazioni dell'algoritmo di scomposizione proposto da Li-Pelejo. 46
4.1 Diagramma di flusso principale dell'implementazione dell'algoritmo di Nielsen-Chuang. 53
4.2 Diagramma di flusso dell'algoritmo per la trasformazione di controlli di tipo '0' in controlli di tipo '1'. 54
4.3 Esempio di trasformazione di controlli da tipo '0' in controlli di tipo '1'. 55
4.4 Diagramma di flusso della scomposizione ricorsiva di Nielsen-Chuang. 56
4.5 Diagramma di flusso della scomposizione di una linea dello schematico secondo l'algoritmo di Nielsen-Chuang. 57
4.6 Diagramma di flusso della procedura di analisi di ogni porta dello schematico A per la scrittura dei circuiti quantistici in OpenQASM 2.0. 58
4.7 Diagramma di flusso per l'implementazione della funzione per la scrittura dei circuiti quantistici in OpenQASM 2.0. 59
4.8 Diagramma di flusso per l'implementazione della funzione per la scrittura di una porta $CU1$ in OpenQASM 2.0. 60
4.9 Diagramma di flusso per l'implementazione della funzione per la scrittura di una porta $CU3$ in OpenQASM 2.0. 62
4.10 Diagramma di flusso per l'implementazione del recupero di fase. . . . 63
4.11 Grafico lineare di confronto del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno. 65
4.12 Grafico logaritmico di confronto del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno. 65
4.13 Grafico con l'interpolazione dei dati del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno, in scala lineare. 66
4.14 Grafico con l'interpolazione dei dati del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno, in scala semi-logaritmica. 66
4.15 Grafico lineare di confronto del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con scomposizione della porta Toffoli quando i test sono abilitati o meno. 68
4.16 Grafico logaritmico di confronto del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con scomposizione della porta Toffoli quando i test sono abilitati o meno. 68

4.17 Grafico con l'interpolazione dei dati del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con scomposizione della porta Toffoli quando i test sono abilitati o meno, in scala lineare. 69
4.18 Grafico con l'interpolazione dei dati del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con scomposizione della porta Toffoli quando i test sono abilitati o meno, in scala semi-logaritmica. 69
4.19 Grafico con l'errore numerico generato durante la scomposizione di Nielsen-Chuang in scala lineare. 70
4.20 Grafico con l'errore numerico generato durante la scomposizione di Nielsen-Chuang in scala semi-logaritmica. 70
4.21 Grafico lineare di confronto del tempo di scrittura dell'OpenQASM e recupero di fase con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno. 72
4.22 Grafico logaritmico di confronto del tempo di scrittura dell'OpenQASM e recupero di fase con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno. 72
4.23 Grafico con l'interpolazione dei dati del tempo di scrittura dell'OpenQASM e recupero di fase con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno, in scala lineare. 74
4.24 Grafico con l'interpolazione dei dati del tempo di scrittura dell'OpenQASM e recupero di fase con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno, in scala semi-logaritmica. . . . 74
4.25 Grafico in scala lineare di confronto del tempo di generazione dell'OpenQASM 2.0 e relativo recupero di fase con scomposizione della porta Toffoli quando i test sono abilitati o meno. 75
4.26 Grafico in scala semi-logaritmica di confronto del tempo di generazione dell'OpenQASM 2.0 e relativo recupero di fase con scomposizione della porta Toffoli quando i test sono abilitati o meno. 75
4.27 Grafico con l'interpolazione dei dati del tempo di generazione dell'OpenQASM 2.0 e relativo recupero di fase con scomposizione della porta Toffoli quando i test sono abilitati o meno, in scala lineare. . . . 76
4.28 Grafico con l'interpolazione dei dati del tempo di generazione dell'OpenQASM 2.0 e relativo recupero di fase con scomposizione della porta Toffoli quando i test sono abilitati o meno, in scala semi-logaritmica. 76
4.29 Grafico con il tempo totale di scomposizione di una matrice unitarie in scala lineare. 78
4.30 Grafico con il tempo totale di scomposizione di una matrice unitarie in scala semi-logaritmica. 78
4.31 Grafico con l'interpolazione del tempo totale di scomposizione di una matrice unitarie, in scala lineare. 79

4.32 Grafico con l'interpolazione del tempo totale di scomposizione di una matrice unitarie, in scala semi-logaritmica. 79
4.33 Grafico con il confronto dell'errore numerico generato dagli algoritmi di scomposizione di Li-Pelejo e Nielsen-Chuang in scala lineare. . . . 81
4.34 Grafico con il confronto dell'errore numerico generato dagli algoritmi di scomposizione di Li-Pelejo e Nielsen-Chuang in scala semi-logaritmica. 81
4.35 Grafico con il confronto del numero di porte quantistiche utilizzato per la descrizione del circuito in OpenQASM 2.0 in caso di abilitazione della scomposizione della porta Toffoli `(toffoli:0)` o meno `(toffoli:1)` in scala lineare. 83
4.36 Grafico con il confronto del numero di porte quantistiche utilizzato per la descrizione del circuito in OpenQASM 2.0 in caso di abilitazione della scomposizione della porta Toffoli `(toffoli:0)` o meno `(toffoli:1)` in scala semi-logaritmica. 83
4.37 Grafico con ilo fitting dei dati del numero di porte quantistiche utilizzato per la descrizione del circuito in OpenQASM 2.0 in caso di abilitazione della scomposizione della porta Toffoli `(toffoli:0)` o meno `(toffoli:1)`, in scala lineare. 84
4.38 Grafico con ilo fitting dei dati del numero di porte quantistiche utilizzato per la descrizione del circuito in OpenQASM 2.0 in caso di abilitazione della scomposizione della porta Toffoli `(toffoli:0)` o meno `(toffoli:1)`, in scala semi-logaritmica. 84
4.39 Diagramma di flusso della tecnica di verifica funzionale del metodo di scomposizione delle matrici unitarie implementato. 87
5.1 Schema a blocchi del circuito della Quantum Phase Estimation (QPE). 92
5.2 Circuito della $QFT^{\dagger}$. 94
5.3 Diagramma di flusso per l'implementazione del metodo di Householder per la creazione di matrici unitarie partendo da vettori unitari. . . 97
5.4 Diagramma di flusso per l'implementazione del metodo Nullspace per la creazione di matrici unitarie partendo da vettori unitari. 98
5.5 Grafico con il tempo di scomposizione, con test disabilitati, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala lineare. 101
5.6 Grafico con il tempo di scomposizione, con test disabilitati, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala semi-logaritmica. 101
5.7 Grafico con il tempo di scomposizione, con test abilitati, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala lineare. 102

5.8 Grafico con il tempo di scomposizione, con test abilitati, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala semi-logaritmica. 102
5.9 Grafico con il numero di porte quantistiche, con disabilitazione della scomposizione della porta Toffoli, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala lineare. 103
5.10 Grafico con il numero di porte quantistiche, con disabilitazione della scomposizione della porta Toffoli, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala semi-logaritmica. 104
5.11 Grafico con il numero di porte quantistiche, con abilitazione della scomposizione della porta Toffoli, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala lineare. 104
5.12 Grafico con il numero di porte quantistiche, con abilitazione della scomposizione della porta Toffoli, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala semi-logaritmica. 105
5.13 Diagramma di flusso nel caso in cui si decida di eseguire più scomposizione per le diverse potenze della matrice CU. 106
5.14 Diagramma di flusso nel caso in cui si decida di stanziare più volte il circuito relativo alla matrice CU^1 per le potenze maggiori di 1 della matrice CU^{2^i}. 107
5.15 Grafico con il tempo di creazione del blocco CU_{rotation}, per test disabilitati, nei casi di multi-scomposizione e di singola-scomposizione, in scala lineare. 109
5.16 Grafico con il tempo di creazione del blocco CU_{rotation}, per test disabilitati, nei casi di multi-scomposizione e di singola-scomposizione, in scala semi-logaritmica. 109
5.17 Grafico con il tempo di creazione del blocco CU_{rotation}, per test abilitati, nei casi di multi-scomposizione e di singola-scomposizione, in scala lineare. 110
5.18 Grafico con il tempo di creazione del blocco CU_{rotation}, per test abilitati, nei casi di multi-scomposizione e di singola-scomposizione, in scala semi-logaritmica. 110
5.19 Grafico con il numero di porte totali del blocco CU_{rotation}, per la scomposizione della porta Toffoli disabilitata, nei casi di multi-scomposizione e di singola-scomposizione, in scala lineare. 111
5.20 Grafico con il numero di porte totali del blocco CU_{rotation}, per la scomposizione della porta Toffoli disabilitata, nei casi di multi-scomposizione e di singola-scomposizione, in scala semi-logaritmica. 111
5.21 Grafico con il numero di porte totali del blocco CU_{rotation}, per la scomposizione della porta Toffoli abilitata, nei casi di multi-scomposizione e di singola-scomposizione, in scala lineare. 112

5.22 Grafico con il numero di porte totali del blocco CU_{rotation}, per la scomposizione della porta Toffoli abilitata, nei casi di multi-scomposizione e di singola-scomposizione, in scala semi-logaritmica. 112
5.23 Diagramma di flusso della tecnica di verifica funzionale dell'implementazione della QPE. 114
6.1 Diagramma di flusso per ottenere la matrice che descrive l'operatore di complemento a due. 121
6.2 Esempio della creazione di una linea matrice del complemento a due. 122
6.3 Grafico con il tempo totale di scomposizione, per test disabilitati, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala lineare. 123
6.4 Grafico con il tempo totale di scomposizione, per test disabilitati, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala semi-logaritmica. 123
6.5 Grafico con il tempo totale di scomposizione, per test abilitati, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala lineare. 124
6.6 Grafico con il tempo totale di scomposizione, per test abilitati, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala semi-logaritmica. 124
6.7 Grafico con il numero totale di porte, nel caso di disabilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala lineare. 125
6.8 Grafico con il numero totale di porte, nel caso di disabilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala semi-logaritmica. 126
6.9 Grafico con il numero totale di porte, nel caso di abilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala lineare. 126
6.10 Grafico con il numero totale di porte, nel caso di abilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala semi-logaritmica. . 127
6.11 Diagramma di flusso per ottenere la matrice che descrive l'operatore di modulo. 130
6.12 Grafico con il tempo totale di scomposizione, per test disabilitati, della matrice rappresentate l'operatore di modulo e la matrice dell'operatore di Ca2, scala lineare. 132
6.13 Grafico con il tempo totale di scomposizione, per test disabilitati, della matrice rappresentate l'operatore di modulo e la matrice dell'operatore di Ca2, scala semi-logaritmica. 132

6.14 Grafico con il tempo totale di scomposizione, per test abilitati, della matrice rappresentate l'operatore di modulo e la matrice dell'operatore di Ca2, scala lineare. 133
6.15 Grafico con il tempo totale di scomposizione, per test abilitati, della matrice rappresentate l'operatore di modulo e la matrice dell'operatore di Ca2, scala semi-logaritmica. 133
6.16 Grafico con il numero totale di porte, nel caso di disabilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore del modulo e quella della Ca2, scala lineare. 134
6.17 Grafico con il numero totale di porte, nel caso di disabilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore del modulo e quella della Ca2, scala semi-logaritmica. . . 134
6.18 Grafico con il numero totale di porte, nel caso di abilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore del modulo e quella della Ca2, scala lineare. 135
6.19 Grafico con il numero totale di porte, nel caso di abilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore del modulo e quella della Ca2, scala semi-logaritmica. 135
6.20 Diagramma di flusso per ottenere la matrice che descrive l'operatore del reciproco. 139
6.21 Grafico con il tempo di scomposizione dell'operatore reciproco, in scala lineare. 140
6.22 Grafico con il tempo di scomposizione dell'operatore reciproco, in scala semi-logaritmica. 141
6.23 Grafico con il numero totale di porte utilizzate nella scomposizione dell'operatore reciproco, in scala lineare. 141
6.24 Grafico con il numero totale di porte utilizzate nella scomposizione dell'operatore reciproco, in scala semi-logaritmica. 142
6.25 Grafico con l'interpolazione del tempo totale di scomposizione degli operatori descritti da look-up table analizzati, in scala lineare. 143
6.26 Grafico con l'interpolazione del tempo totale di scomposizione degli operatori descritti da look-up table analizzati, in scala semi-logaritmica. 143

Capitolo 1

Introduzione al Quantum Computing

In questo capitolo si riporta una breve introduzione al quantum computing ed al suo formalismo. Per ulteriori approfondimenti si invita a fare riferimento a [1, 2].

1.1 Il Qubit

L'unità di informazione del quantum computing è il **qubit** (quantum bit). Esso è codificato su una grandezza fisica quantistica, quale lo spin di un elettrone o del nucleo [3] di un atomo o su parametri elettrici quantizzati, nel caso di qubit superconduttivi [4]. Per questo motivo, la caratteristica peculiare del qubit è che il suo stato è descritto secondo il principio di sovrapposizione della meccanica quantistica, per cui esso è sempre descritto da una combinazione lineare di autostati da esso assumibili dopo una misurazione (per esempio nel caso del qubit codificato su spin-$\frac{1}{2}$, i due autostati sono *spin-up* e *spin-down*). Gli autostati del qubit sono denominati $|0\rangle$ e $|1\rangle$ e ciascuno di questi può essere descritto, in accordo con la notazione di Dirac, come un vettore colonna di due elementi:

$$|0\rangle = \begin{matrix}\mathbf{0}\\ \mathbf{1}\end{matrix}\begin{bmatrix}1\\0\end{bmatrix} \qquad |1\rangle = \begin{matrix}\mathbf{0}\\ \mathbf{1}\end{matrix}\begin{bmatrix}0\\1\end{bmatrix} . \tag{1.1}$$

Una peculiarità di questi due vettori è che costituiscono una base ortonormale.

Infatti, entrambi hanno norma pari a 1 e il loro prodotto scalare (prodotto *bra-ket*, secondo la notazione di Dirac) è nullo. Si riporta un esempio di seguito:

$$\langle 0|1\rangle = \langle 0| \cdot |1\rangle = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0 + 0 = 0 \tag{1.2}$$

Lo stato generico di un qubit può essere descritto da una combinazione lineare di questi autovettori:

$$|\psi\rangle = \alpha_0 \cdot |0\rangle + \alpha_1 \cdot |1\rangle = \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} , \tag{1.3}$$

dove α_0, $\alpha_1 \in \mathbb{C}$ è sono le ampiezze di probabilità dello stato $|\psi\rangle$ di assumete rispettivamente gli stati $|0\rangle$ o $|1\rangle$, dopo una misurazione.

Da un punto di vista pratico, misurare lo stato di un qubit consiste nel cambiarne lo stato da uno di sovrapposizione ad uno degli autostati della base ($|0\rangle$ o $|1\rangle$), secondo le probabilità:

$$\begin{aligned} \mathbb{P}(|0\rangle) &= |\alpha_0|^2 \\ \mathbb{P}(|1\rangle) &= |\alpha_1|^2 \end{aligned} \tag{1.4}$$

Assumendo che gli unici stati in cui il qubit può essere misurato sono $|0\rangle$ e $|1\rangle$, vale la seguente relazione:

$$|\alpha_0|^2 + |\alpha_1|^2 = 1 \, . \tag{1.5}$$

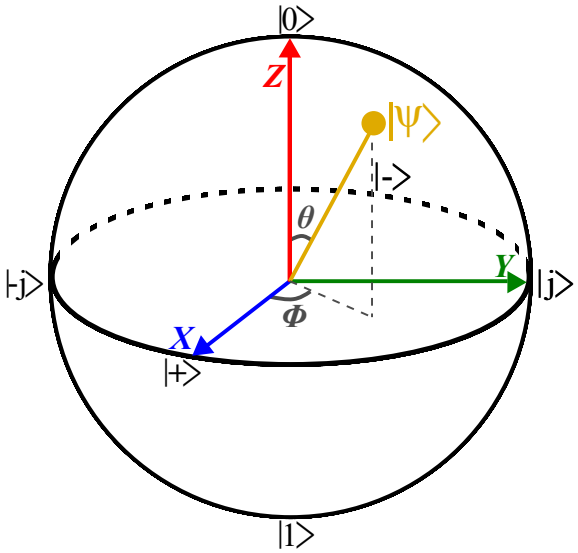


Figura 1.1: Sfera di Bloch.

Lo stato di un singolo qubit è solitamente rappresentato con la **sfera di Bloch** (Figura 1.1). Da un punto di vista visivo-geometrico, ogni vettore di stato ha lunghezza unitaria, ha origine nell'origine degli assi cartesiani e termina in uno dei punti sulla superficie della sfera. Ogni operazione sul singolo qubit corrisponde ad uno spostamento del vettore di stato del qubit sulla superficie della sfera di Bloch. Lo stato di un qubit può essere descritto in funzione di due angoli caratteristici θ e ϕ, secondo la seguente formula:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{j\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ e^{j\phi}\sin\left(\frac{\theta}{2}\right) \end{bmatrix}. \tag{1.6}$$

Gli stati $|0\rangle$, $|1\rangle$, $|+\rangle$, $|-\rangle$, $|j\rangle$ e$|-j\rangle$, che possono essere visti come gli *stati notevoli* del singolo qubit, si trovano sui punti della sfera di Bloch giacenti sugli assi z, x e y ripetitivamente.

1.2 Sistemi con più qubit

In un contesto applicativo, è necessario adoperare almeno due qubit. Gli stati base di un sistema con due qubit sono ottenuti combinando gli stati base per ogni singolo qubit attraverso il prodotto tensoriale (o di Kronecker), che viene indicato con il

simbolo $\otimes$:

$$
\begin{aligned}
|0\rangle \otimes |0\rangle &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = |00\rangle \ , \\
|0\rangle \otimes |1\rangle &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = |01\rangle \ , \\
|1\rangle \otimes |0\rangle &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle \ , \\
|1\rangle \otimes |1\rangle &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |11\rangle \ .
\end{aligned}
\tag{1.7}
$$

Lo stato di sovrapposizione di due qubit separati può essere generalmente scritto come:

$$
|\psi_1\rangle \otimes |\psi_2\rangle = \begin{bmatrix} \alpha_{10} \\ \alpha_{11} \end{bmatrix} \otimes \begin{bmatrix} \alpha_{20} \\ \alpha_{21} \end{bmatrix} = \begin{bmatrix} \alpha_{10} \cdot \alpha_{20} \\ \alpha_{10} \cdot \alpha_{21} \\ \alpha_{11} \cdot \alpha_{20} \\ \alpha_{11} \cdot \alpha_{21} \end{bmatrix} = \begin{matrix} \mathbf{00} \\ \mathbf{01} \\ \mathbf{10} \\ \mathbf{11} \end{matrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \ , \tag{1.8}
$$

dove $a, b, c, d \in \mathbb{C}$ e $|a|^2 + |b|^2 + |c|^2 + |d|^2 = 1$, e sono rispettivamente le probabilità degli stati $|00\rangle$, $|01\rangle$, $|10\rangle$ e $|11\rangle$. Si riporta di seguito la generalizzazione con n-qubit:

$$
|\psi_1\rangle \otimes |\psi_2\rangle \ldots \otimes |\psi_n\rangle = \begin{matrix} \mathbf{0\ldots00} \\ \mathbf{0\ldots01} \\ \vdots \\ \mathbf{1\ldots11} \end{matrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \end{bmatrix} \ . \tag{1.9}
$$

dove $N = 2^n$ e $\sum_{i=0}^{N-1} |a_i| = 1$. Si ritiene doveroso precisare che lo stato di un sistema

con n qubit può essere sempre descritto da un vettore colonna di lunghezza 2^n, ma non è garantito che lo stato sia ottenibile attraverso prodotti tensoriali di singoli qubit. Un esempio notevole è il seguente, coinvolgente due qubit:

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} . \tag{1.10}$$

Non è possibile descrivere questo stato come il prodotto tensoriale di due stati, uno per qubit. Un'altra peculiarità di questo stato è che se si misura $|0\rangle$ sul qubit destro, si è certi che il sinistro varrà $|0\rangle$, viceversa se si misura $|1\rangle$ sul qubit destro quello sinistro varrà sicuramente $|1\rangle$. Questo è il più noto esempio di stato soggetto ad **entanglement**, una correlazione intrinseca tra sotto-sistemi quantistici che rende impossibile la loro analisi individuale.

I principi di sovrapposizione ed entanglement, insieme al fenomeno dell'interferenza, sono i principali elementi che consentono al quantum computing di risolvere alcuni problemi complessi, per esempio quelli di ricerca [5] e di fattorizzazione degli interi [6], con costi computazionali minori dei migliori corrispettivi algoritmi classici. In particolare, con il principio di sovrapposizione degli stati la stessa operazione può essere valutata simultaneamente su più dati, ciascuno dei quali associato ad uno stato quantistico, l'interferenza è adoperata per incrementare la probabilità della soluzione del problema, mentre l'entanglement tra i qubit può facilitare la convergenza alla soluzione del problema, nel senso che se si esegue una misurazione su uno o più qubit, il risultato influenza istantaneamente i corrispettivi valori degli altri qubit soggetti all'interazione.

1.3 Matrici Hermitiane e Unitarie

In questa sezione sono introdotti i concetti di matrici unitarie e matrici hermitiane, evidenziandone alcune proprietà e le loro differenze. Entrambe sono generalmente complesse.

Tutte le matrici Hamiltoniane che descrivono i sistemici fisici quantistici, quindi l'hardware dei computer quantistici, sono hermitiane, mentre le matrici unitarie descrivono le evoluzioni associate ad un sistema quantistico isolato. In sostanza, un'evoluzione unitaria è la risoluzione di un'equazione di Schrödinger associata ad una matrice Hamiltoniana . In particolare, nel caso di matrice hermitiana H stazionaria (dimensionata in unità di misura di frequenza), l'evoluzione unitaria si può scrivere come [2]:

$$U(t) = e^{j2\pi Ht}\,. \tag{1.11}$$

È evidente l'esistenza di un'intrinseca correlazione tra matrici hermitiane ed unitarie. Un esempio concreto, in cui la relazione tra queste matrici è adoperata per finalità applicative, è l'algoritmo di Quantum Phase Estimation (QPE), descritto nel capitolo 5.
Si dice matrice **unitaria** una matrice per la quale vale la seguente relazione:

$$U \cdot U^{\dagger} = U^{\dagger} \cdot U = \mathbf{I}\,, \tag{1.12}$$

dove: U è una matrice quadrata di ordine N, $U^{\dagger}$ è la matrice trasposta coniugata di U e $\mathbf{I}$ è la matrice identità.
Dalla definizione di matrice unitaria si può constatare che:

$$U^{\dagger} = U^{-1}\,. \tag{1.13}$$

Si dice matrice **hermitiana** una matrice tale che:

$$H = H^{\dagger}\,, \tag{1.14}$$

dove H è una matrice quadrata di ordine N e $H^{\dagger}$ è la matrice trasposta e coniugata di H.
Si riportano le seguenti proprietà di una matrice hermitana:

- Se $H \in \mathbb{R}^N$: la matrice hermitana H è simmetrica.
- se $H \in \mathbb{C}^N$: la triangolare superiore di H è la complessa coniugata della triangolare inferiore;

- Una matrice hermitiana ha autovalori reali.

È importante evidenziare che **non è garantito che**:

$$H^\dagger = H^{-1} \longrightarrow H^\dagger \cdot H = \mathbf{I}\,, \tag{1.15}$$

per cui una matrice hermitiana non è necessariamente anche unitaria e viceversa.

Le matrici di Pauli hanno la particolarità di essere sia matrici hermitiane che unitarie:

$$X = X^\dagger = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\,, \tag{1.16}$$

$$Y = Y^\dagger = \begin{bmatrix} 0 & -j \\ j & 0 \end{bmatrix}\,, \tag{1.17}$$

$$Z = Z^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\,. \tag{1.18}$$

Si può notare come per tutte e tre le matrici valga: $X \cdot X^\dagger = \mathbf{I}$, $Y \cdot Y^\dagger = \mathbf{I}$ e $Z \cdot Z^\dagger = \mathbf{I}$.

Da una matrice hermitiana è possibile ricavare la matrice unitaria associata come:

$$U = e^{\frac{2\pi j}{2^n} \cdot H}\,, \tag{1.19}$$

dove n è il numero di *ancilla* qubit come spiegato meglio nel capitolo 5.

Per informazioni più approfondite sulle matrici unitarie ed hermitiane fare riferimento a [1].

1.4 Introduzione alle principali porte quantistiche

In questa sezione sono descritte le principali porte quantistiche utilizzate nella descrizione dei circuiti in OpenQASM 2.0 dal metodo di scomposizione implementato

in questa tesi.

Le porte quantistiche sono usate per modificare lo stato di uno o più qubit secondo la matrice unitaria ad essa associata. Nel caso di un singolo qubit, tale manipolazione corrisponde ad uno spostamento del vettore rappresentate lo stato del qubit all'interno della sfera di Bloch ed è descritta da una matrice 2×2. Una porta che agisce su n qubit ha una matrice unitaria di dimensioni $2^n \times 2^n$. In un sistema a m qubit, il prodotto tensoriale permette di valutare l'effetto di una porta a $n < m$ qubit sullo stato complessivo del sistema. Per esempio, dato un sistema a tre qubit $|q\rangle = |q_2\rangle\,|q_1\rangle\,|q_0\rangle$, l'effetto di una porta a singolo-qubit generica U su $|q_1\rangle$ può essere scritta come:

$$U\,|q\rangle = \mathbf{I} \otimes U \otimes \mathbf{I}\,|q\rangle\ , \tag{1.20}$$

dove $\mathbf{I}$ è la matrice identità di ordine 2. Per maggiori informazioni sulle porte quantistiche, si rimanda a [1, 2].

In questa tesi è utilizzata la notazione **IBM** per rappresentare l'ordine dei qubit, per cui il qubit più significativo (*Most Significant Qubit* o MSQ) sarà quello più in basso mentre quello meno significativo (*Least Significant Qubit* o LSQ) sarà quello più in alto.

$$\begin{array}{rc} |LSQ\rangle & \text{——} \\ \vdots & \\ |MSQ\rangle & \text{——} \end{array} \tag{1.21}$$

1.4.1 Porta di Pauli X

La porta di Pauli X (di seguito chiamata semplicemente porta X) è il corrispettivo quantistico della porta NOT nella computazione classica.
La porta X agisce su un singolo qubit, è permette di passare dallo stato $|0\rangle$ allo stato $|1\rangle$ e viceversa.

La matrice unitaria associata è:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} . \tag{1.22}$$

Il simbolo utilizzato per rappresentarla nei circuiti è:

$$|q_0\rangle \text{ —}\oplus\text{—} \tag{1.23}$$

Considerando la matrice unitaria associata si può osservare che:

$$X \cdot |0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \tag{1.24}$$

$$X \cdot |1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \tag{1.25}$$

In generale per uno stato di ingresso $|\psi_i\rangle$ l'uscita corrisponderà a $|\psi_o\rangle$ secondo:

$$X \cdot |\psi_i\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_0 \end{bmatrix} = |\psi_o\rangle \tag{1.26}$$

In pratica si scambiano le probabilità legate agli stati $|0\rangle$ e $|1\rangle$.
Per quanto concerne la sfera di Bloch, la porta X corrisponde ad una rotazione di 180° dell'asse x su se stesso. Per cui l'effetto è quello mostrato in Figura 1.2.

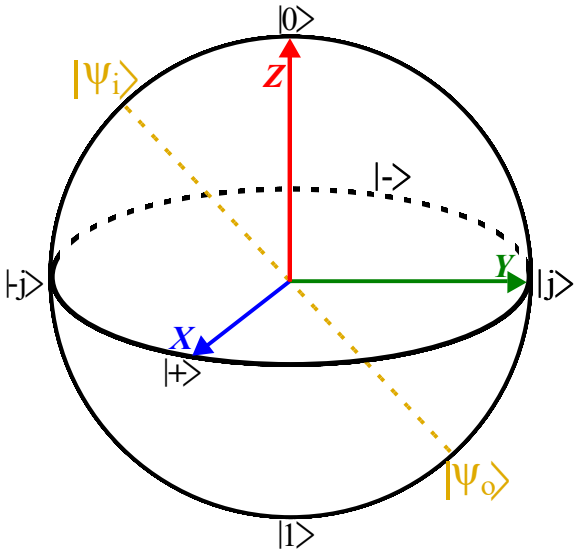


Figura 1.2: Rotazione sulla sfera di Bloch della porta X.

1.4.2 Porta CX

La porta CX è la versione controllata della porta X descritta nella sezione precedente.
Il circuito è il seguente:

$$\begin{array}{l} |q_0\rangle \\ |q_1\rangle \end{array} \quad \begin{array}{l} \text{—}\oplus\text{—} \\ \text{—}\bullet\text{—} \end{array} \quad \begin{array}{l} |q_0 \oplus q_1\rangle \\ |q_1\rangle \end{array} \tag{1.27}$$

In questo caso la rotazione viene applicata al qubit q_0 se e solo se lo stato del qubit q_1 è uguale a $|1\rangle$.
Il qubit q_1 prende il nome di **qubit di controllo** mentre il qubit q_0 prende il nome di **qubit target**.
Considerando la notazione di IBM la matrice associata al circuito precedente è:

$$CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{1.28}$$

Si può osservare che:

$$CX \cdot |00\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = |00\rangle \tag{1.29}$$

$$CX \cdot |01\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = |01\rangle \tag{1.30}$$

$$CX \cdot |10\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |11\rangle \tag{1.31}$$

$$CX \cdot |11\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle \tag{1.32}$$

In generale per uno stato di ingresso $|\psi_i\rangle$ l'uscita corrisponderà a $|\psi_o\rangle$ secondo:

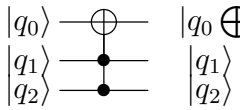
$$CX \cdot |\psi_i\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_3 \\ \alpha_2 \end{bmatrix} = |\psi_o\rangle \tag{1.33}$$

In pratica si scambiano le ampiezze di probabilità associate agli stati $|10\rangle$ e $|11\rangle$.

1.4.3 Porta CCX

La porta CCX è la versione con due controlli della porta X, Essa è chiamata porta Toffoli.

Il circuito quantistico è:



(1.34)

In questo caso la rotazione viene applicata al qubit q_0 se e solo se sia lo stato del qubit q_1 che quello di q_2 sono entrambi uguali a $|1\rangle$.
I qubit q_1 e q_2 prendono il nome di **qubit di controllo** mentre il qubit q_0 prende il nome di **qubit target**.
Considerando la notazione di IBM la matrice associata al circuito precedente è:

$$CCX = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} . \tag{1.35}$$

Si può osservare che:

$$CCX \cdot |000\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = |000\rangle \tag{1.36}$$

$$CCX \cdot |110\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |111\rangle \tag{1.37}$$

$$CCX \cdot |111\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |110\rangle \tag{1.38}$$

In generale per uno stato di ingresso $|\psi_i\rangle$ l'uscita corrisponderà a $|\psi_o\rangle$ secondo:

$$CCX \cdot |\psi_i\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \end{bmatrix} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_7 \\ \alpha_6 \end{bmatrix} = |\psi_o\rangle \tag{1.39}$$

In pratica si scambiano le ampiezze di probabilità associate agli stati $|110\rangle$ e $|111\rangle$.

Porta C^nX In generale esistono versioni controllate della porta X con n qubit di controllo: C^nX. L'applicazione della porta X sul qubit target è effettuata se e solo se lo stato di tutti i qubit di controllo è uguale a $|1\rangle$

1.4.4 Porta U1

La porta $U1$ è una porta agente su un singolo qubit, che effettua una rotazione generica, con angolo λ, intorno all'asse z del vettore di stato del qubit.
Il circuito è:

$$|q_0\rangle \text{ —} \boxed{U1(\lambda)} \text{—} \tag{1.40}$$

La matrice unitaria associata è:

$$U1(\lambda) = \begin{bmatrix} 1 & 0 \\ 0 & e^{j\lambda} \end{bmatrix} . \tag{1.41}$$

Sulla sfera di Bloch la rotazione U1 può essere rappresentata come mostrato in Figura 1.3.

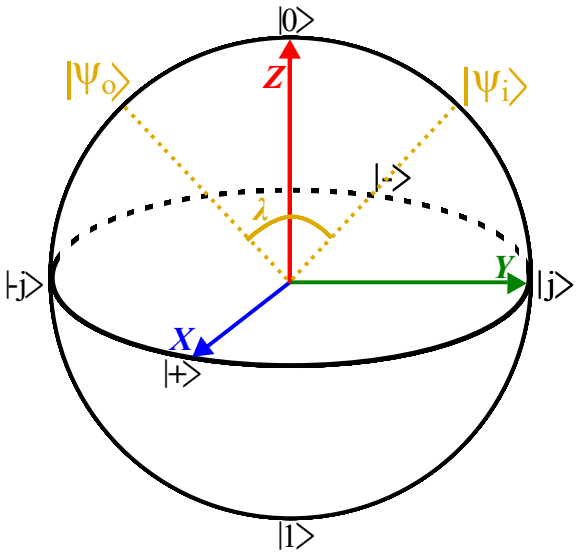


Figura 1.3: Rotazione sulla sfera di Bloch della porta $U1$.

Si può osservare che:

$$U1(\lambda) \cdot |0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{j\lambda} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |0\rangle \tag{1.42}$$

$$U1(\lambda) \cdot |1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{j\lambda} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ e^{j\lambda} \end{bmatrix} \tag{1.43}$$

In generale per uno stato di ingresso $|\psi_i\rangle$ l'uscita corrisponderà a $|\psi_o\rangle$ secondo:

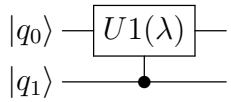
$$U1(\lambda) \cdot |\psi_i\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{j\lambda} \end{bmatrix} \cdot \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \cdot e^{j\lambda} \end{bmatrix} = |\psi_o\rangle \tag{1.44}$$

Esistono dei casi notevoli di angolo λ:

- $\lambda = \pi$: $U1(\pi) = Z$ (porta di Pauli Z);
- $\lambda = \frac{\pi}{2}$: $U1(\frac{\pi}{2}) = S$;
- $\lambda = \frac{\pi}{4}$: $U1(\frac{\pi}{4}) = T$;

1.4.5 Porta CU1

La porta $CU1$ è la versione controllata della porta $U1$ descritta nella sotto-sezione precedente.
Il circuito è:



(1.45)

L'applicazione della porta $U1$ sul qubit target (q_0) è effettuata se e solo se lo stato del qubit di controllo (q_1) è uguale a $|1\rangle$.
Considerando la notazione di IBM la matrice associata al circuito precedente è:

$$CU1(\lambda) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{j\lambda} \end{bmatrix}. \tag{1.46}$$

In generale per uno stato di ingresso $|\psi_i\rangle$ l'uscita corrisponderà a $|\psi_o\rangle$ secondo:

$$CU1(\lambda) \cdot |\psi_i\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{j\lambda} \end{bmatrix} \cdot \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \cdot e^{j\lambda} \end{bmatrix} = |\psi_o\rangle \tag{1.47}$$

1.4.6 Porta U3

La porta $U3$ è una porta agente su un singolo qubit, che effettua una rotazione generica, comprendente gli angoli θ, ϕ e λ, intorno all'asse x, y e z rispettivamente del vettore di stato del qubit.
Il circuito è:

$$|q_0\rangle \text{ —} \boxed{U3(\theta, \phi, \lambda)} \text{—} \tag{1.48}$$

La matrice unitaria associata è:

$$U3(\theta, \phi, \lambda) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{j\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{j\phi}\sin\left(\frac{\theta}{2}\right) & e^{j(\lambda+\phi)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix}. \tag{1.49}$$

In generale per uno stato di ingresso $|\psi_i\rangle$ l'uscita corrisponderà a $|\psi_o\rangle$ secondo:

$$\begin{aligned} U3(\theta, \phi, \lambda) \cdot |\psi_i\rangle &= \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{j\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{j\phi}\sin\left(\frac{\theta}{2}\right) & e^{j(\lambda+\phi)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix} \cdot \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \\ &= \begin{bmatrix} \alpha_0 \cdot \cos\left(\frac{\theta}{2}\right) + \alpha_1 \cdot \left(-e^{j\lambda}\sin\left(\frac{\theta}{2}\right)\right) \\ \alpha_0 \cdot e^{j\phi}\sin\left(\frac{\theta}{2}\right) + \alpha_1 \cdot e^{j(\lambda+\phi)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix} = |\psi_o\rangle \end{aligned} \tag{1.50}$$

1.4.7 Porta CU3

La porta $CU3$ è la versione controllata della porta $U3$ descritta nella sotto-sezione precedente.
Il circuito è:

$$\begin{aligned} |q_0\rangle &\text{ —} \boxed{U3(\theta, \phi, \lambda)} \text{—} \\ |q_1\rangle &\text{ ———●———} \end{aligned} \tag{1.51}$$

L'applicazione della porta $U3$ sul qubit target (q_0) è effettuata se e solo se lo stato del qubit di controllo (q_1) è uguale a $|1\rangle$.

Considerando la notazione di IBM la matrice associata al circuito precedente è:

$$CU3(\theta,\phi,\lambda) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\left(\frac{\theta}{2}\right) & -e^{j\lambda}\sin\left(\frac{\theta}{2}\right) \\ 0 & 0 & e^{j\phi}\sin\left(\frac{\theta}{2}\right) & e^{j(\lambda+\phi)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix}. \tag{1.52}$$

In generale per uno stato di ingresso $|\psi_i\rangle$ l'uscita corrisponderà a $|\psi_o\rangle$ secondo:

$$\begin{aligned} CU3(\theta,\phi,\lambda)\cdot|\psi_i\rangle &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\left(\frac{\theta}{2}\right) & -e^{j\lambda}\sin\left(\frac{\theta}{2}\right) \\ 0 & 0 & e^{j\phi}\sin\left(\frac{\theta}{2}\right) & e^{j(\lambda+\phi)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix} \cdot \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \\ &= \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2\cdot\cos\left(\frac{\theta}{2}\right) + \alpha_3\cdot\left(-e^{j\lambda}\sin\left(\frac{\theta}{2}\right)\right) \\ \alpha_2\cdot e^{j\phi}\sin\left(\frac{\theta}{2}\right) + \alpha_3\cdot e^{j(\lambda+\phi)}\cos\left(\frac{\theta}{2}\right)e^{j\lambda} \end{bmatrix} = |\psi_o\rangle \end{aligned} \tag{1.53}$$

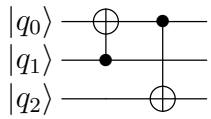
Per maggiori informazioni su queste e altre porte quantistiche fare riferimento a [1].

Capitolo 2

OpenQASM 2.0 e descrizione generale dell'infrastruttura sviluppata

Il paradigma di quantum computing *general-purpose* maggiormente consolidato è quello dei **circuiti quantistici** [2], che consiste nella definizione di una sequenza di operazioni, in termini di porte e/o matrici unitarie, che modificano lo stato dei qubit secondo l'algoritmo da implementare.
Oggigiorno non è ancora disponibile uno standard per la descrizione e la progettazione di questi circuiti. Tuttavia, il linguaggio **OpenQASM 2.0** [7], proposto da IBM nel 2017, che nasce come linguaggio di interfaccia per automatizzare la costruzione di circuiti quantistici da eseguire sul proprio hardware accessibile via-*cloud*, si è ampiamente diffuso nel modo del quantum computing da diventare uno standard *de-facto*, adottato da più framework per la progettazione e l'esecuzione di algoritmi su hardware reale o simulatori.
Il linguaggio OpenQASM 2.0 ha una sintassi che eredità caratteristiche dal linguaggio *Verilog*, come per esempio il punti e virgola alla fine di una linea di codice oppure i commenti introdotti da "//".
Usando l'OpenQASM 2.0 è possibile descrive i circuiti quantistici in maniera *strutturale*, cioè andando a descrivere le singole porte e i loro collegamenti all'interno del

circuito. Per esempio il seguente circuito:



(2.1)

Può essere descritto utilizzando l'OpenQASM 2.0 come:

```
cx q1, q0;
cx q0, q2;
```

Dove il primo elemento di ogni linea è la porta quantistica che si vuole costruire, mentre gli altri elementi della linea sono la lista di qubit su cui la porta agisce. Nell'esempio riportato, `cx` rappresenta la porta X controllata ed i due elementi successivi a `cx` su ciascuna riga sono rispettivamente i qubit di controllo e target della porta.

In OpenQASM 2.0 è possibile racchiudere un circuito quantistico, in una porta "*custom*" per poter essere riutilizzato all'interno di un altro circuito più complesso. Continuando l'esempio precedente è possibile definire una porta come riportato di seguito:

```
gate my_gate q0, q1, q2 {
    cx q1, q0;
    cx q0, q2;
}
```

dove `my_gate` è il nome della porta "*custom*" appena definita. La definizione di tali porte deve essere salvata all'interno di file con estensione `.inc`, per poter essere adoperate in altri circuiti. Per riutilizzare, la porta appena definita, all'interno di un altro circuito, è possibile richiamare il file tramite l'istruzione `include`, per poi riutilizzare la porta "*custom*" richiamandone semplicemente il nome ed indicando la lista di qubit con cui lavorerà come segue:

```
include "my_gate.inc";

gate my_second_gate q0, q1, q2, q3 {
    my_gate q0, q3, q1;
    x q2;
    cu1(1.57) q2, q0;
}
```

In questo modo è possibile realizzare circuiti complessi, in modo gerarchico, dividendo la complessità totale sulla realizzazione dei vari blocchi elementari che lo compongono.
Il livello gerarchicamente più alto del circuito deve essere inserito all'interno di un file con estensione `.qasm` che potrà essere eseguito da simulatori oppure su un hardware reale. Tale file, presenta alcune particolarità sintattiche: la prima di linea (escluso eventuali linee di commento) del file `.qasm` dovrà essere per forza la linea indicante la versione dell'OpenQASM che si vuole utilizzare, in questo caso `OpenQASM 2.0`, all'interno di tali file è possibile definire registri di qubit tramite il comando `qreg`, oppure definire dei registri di bit con il comando `creg` su cui riportare i risultati della misura dei qubit alla fine dell'algoritmo tramite il comando `measure`.

```
OPENQASM 2.0;

include "my_second_gate.inc";

qreg q[4];
creg c[4];

my_second_gate q[3], q[0], q[1], q[2];
x q[3];
cu3(1.57, 0, 3.14) q[1], q[0];

measure q -> c;
```

Per tutte le altre regole sintattiche dell'OpenQASM 2.0 fare riferimento a [7].

2.1 Limiti dell'OpenQASM 2.0

Nonostante la sua versatilità l'OpenQASM 2.0 presenta alcuni limiti, che rendono più ardua la descrizione di circuiti complessi.

Il primo limite è legato al fatto che l'unica modalità di descrizione di circuiti quantistici è quella strutturale, per cui è necessario descrivere le singole porte e i singoli collegamenti per ogni elemento del circuito. Anche sfruttando la costruzione gerarchica dei circuiti, il livello di complessità dell'implementazione di alcuni tipi di

algoritmi quantistici potrebbe risultare elevato. Quello che si vorrebbe avere è la possibilità di descrivere a livello *comportamentale* l'algoritmo quantistico, analogamente a quanto avviene con i linguaggi di descrizione dell'hardware nel calcolo classico. A livello quantistico la descrizione comportamentale consiste nella realizzazione della matrice unitaria ad esso associata, pertanto quello che si desidererebbe avere è un infrastruttura che sia in grado di fornire la descrizione strutturale di un circuito a partire dalla matrice unitaria.

Il secondo limite è legato al fatto che nel linguaggio OpenQASM 2.0 possono essere descritte porte: agenti su un singolo qubit, di tipo $singolo-controllo\ singolo-target$ ovvero porte su due qubit con un qubit di controllo e un qubit target e la porta Toffoli, con due controlli e un target. Questa limitazione complica la descrizione di circuiti con porte con più di un controllo e un singolo target ($multi-controllo\ singolo-target$) ed anche la descrizione di porte ($singolo-controllo\ multi-target$) adoperate in molti algoritmi quantistici, come per esempio nell'algoritmo della Quantum Phase Estimation (QPE) mostrata nel capitolo 5.

2.2 Soluzione proposta

All'interno del laboratorio VLSI del Politecnico di Torino è in fase di sviluppo una libreria Python per la generazione di circuiti descritti in OpenQASM 2.0, indipendente da qualsiasi tipo di piattaforma proprietaria, adoperabili in un contesto di progettazione modulare, in cui i circuiti di libreria costituiscono dei *sotto-componenti* di circuiti quantistici più complessi eseguibili su *backend* compatibili con questo linguaggio. La libreria è stata sviluppata a partire da un precedente lavoro di tesi contenente soli circuiti aritmetici [8].

L'obiettivo di questa tesi è stato quello di integrare nella libreria un'infrastruttura in grado di ottenere data in ingresso una generica matrice unitaria U il corrispondente circuito quantistico, descritto in OpenQASM 2.0, attraverso l'esecuzione di diversi step di scomposizione e traduzione.

L'algoritmo di scomposizione, è stato sviluppato interamente in **Python** e comprende tre passi principali, come mostrato in Figura 2.1.

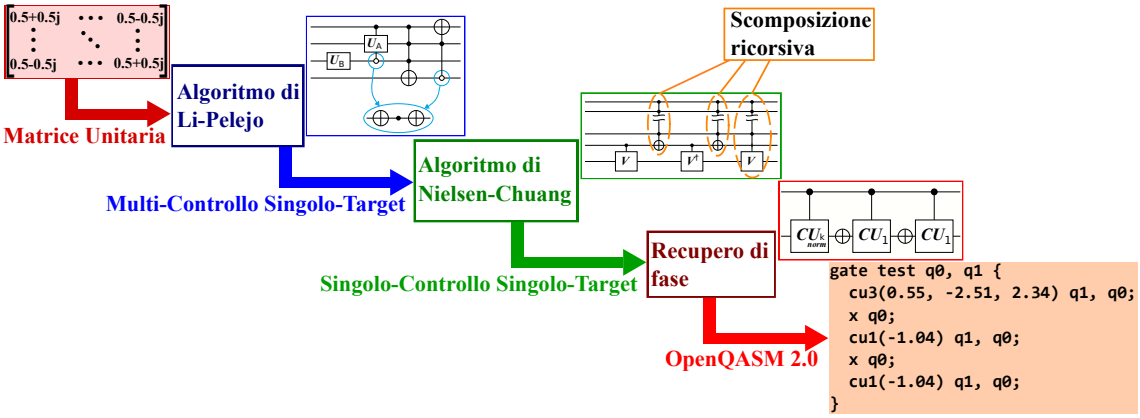


Figura 2.1: Schema a blocchi raffigurante i tre passi di scomposizione dell'algoritmo implementato.

Il primo consiste nell'applicare il metodo di scomposizione delle matrici unitarie proposto da Chi Kwong Li e Diane Christine Pelejo [9], ottenendo in uscita una sequenza di matrici equivalente ad U. L'algoritmo è di tipo iterativo e prevede l'individuazione di r matrici unitarie U_k tali che $U_r U_{r-1} \cdots U_1 U = I$, ottenendo di conseguenza $U = U_1^\dagger \cdots U_r^\dagger$. In particolare, ad ogni iterazione k dell'algoritmo, una coppia di elementi fuori-diagonale della matrice residua $U_{k-1} \cdots U_1 U$ (uno sulla triangolare inferiore, l'altro sulla superiore simmetrico al precedente) è eliminata da U_k, rendendo così il residuo gradualmente una matrice identità. Ciascuna $U_k^\dagger$ è associata ad una porta **singolo qubit** o **multi-controllo singolo-target** (riquadro blu in Figura 2.1).
Poiché in OpenQASM 2.0 sono supportate solo porte controllate di tipo singolo-controllo singolo-target, il secondo passo dell'algoritmo prevede una **scomposizione ricorsiva** delle porte multi-controllo singolo-target, descritta da Michael A. Nielsen e Isaac L. Chuang in [2], al fine di ottenere il formato delle porte desiderato (come mostrato nel riquadro verde in Figura 2.1).

La scomposizione ricorsiva non è da sola sufficiente a descrivere il circuito in OpenQASM 2.0, in quanto essa restituisce porte controllate con l'evoluzione unitaria del

target di tipo:

$$U_{\text{target}} = \begin{bmatrix} |U_{00}|e^{i\phi_{00}} & |U_{01}|e^{i\phi_{01}} \\ |U_{10}|e^{i\phi_{10}} & |U_{11}|e^{i\phi_{11}} \end{bmatrix},$$

non corrispondente alle porte U_1 e U_3 del linguaggio, aventi U_{00} reale. L'ultimo passo di scomposizione, chiamato **recupero di fase**, si occupa della costruzione OpenQASM 2.0-compatibile di tutte le porte del circuito, sostituendo una porta CU_{target} con una sequenza equivalente costituita da cinque porte di tipo X, CU_1 e CU_3 (come mostrato nel riquadro rosso in Figura 2.1), tenenti traccia di ϕ_{00}.

Da un punto di vista funzionale, l'infrastruttura software genera dei file `.inc` contenenti le porte *custom* corrispondenti alle matrici unitarie di ingresso. Inoltre, se è necessario generare dei file direttamente utilizzabili, essa dispone di funzioni che creano dei file "contenitori" di tipo `.qasm` richiamanti le porte definite all'interno dei file `.inc` ed occupantesi della mappatura dei vari qubit sugli ingressi della porta *custom*. Questo file `.qasm` risulta estremamente importante per la verifica funzionale della scomposizione di una matrice unitaria U, riportata in Sezione 4.4.

La scomposizione è effettuata tenendo conto di alcune impostazioni di esecuzione configurabili dall'utente. In particolare, è possibile scegliere se abilitare i test (flag `testEN`), nel caso in cui si abbia la necessità di verificare l'unitarietà delle matrici all'interno di una certa tolleranza (anch'essa impostabile dall'utente tramite la variabile `toll`), e se scomporre la porta di Toffoli (flag `toffoli`), che come già espresso è l'unica porta multi-controllo singolo-target disponibile in OpenQASM 2.0.

Il metodo si scomposizione proposto è prototipale e concepito per ottenere un circuito in OpenQASM 2.0 funzionalmente valido. Nei capitoli successivi sono riportati solamente alcuni esempi illustrativi di verifica funzionale, ma si ritiene necessario precisare in questa sede che questi test sono stati effettuati per tutti i circuiti esaminati e che hanno tutti fornito un esito positivo. Inoltre, l'infrastruttura software è stata sviluppata per garantire l'integrazione futura di algoritmi di scomposizione delle matrici unitarie diversi da quello descritto in Figura 2.1. Pertanto il metodo proposto in questa tesi vuole fungere da blocco di riferimento per altri algoritmi implementabili in futuro.

Capitolo 3

Scomposizione di Li-Pelejo

In questo capitolo viene illustrato il metodo di scomposizione di matrici unitarie proposto da Chi Kwong Li e Diane Christine Pelejo nell'articolo [9].
In particolare le sezioni riportano:

- La teoria alla base del metodo di scomposizione;
- La spiegazione dell'implementazione in Python;
- I risultati ottenuti per alcune scomposizioni, evidenziandone le performance in termini di tempo impiegato ed errore numerico generato.

3.1 Teoria

Ad ogni circuito quantistico su n qubit si può associare una matrice unitaria U di dimensioni NxN ($U \in M_N$), dove $N = 2^n$. Dualmente data una matrice unitaria U è possibile ricavare un circuito quantistico ad essa "equivalente". Esistono diversi modi per scomporre una matrice unitaria; in questa tesi è stato analizzato l'algoritmo proposto da Li e Pelejo, che nel seguito sarà denominato metodo/scomposizione di Li-Pelejo o algoritmo di Li-Pelejo.

Questo metodo si basa sull'assunzione che: data una matrice unitaria $U \in M_N$, si possono trovare r matrici unitarie tali che:

$$U_r \cdots U_1 U = \mathbf{I}_N \,, \tag{3.1}$$

dove con $\mathbf{I}_N$ si indica la matrice identità di dimensioni NxN e:

$$r \leq \frac{N \cdot (N-1)}{2} \,. \tag{3.2}$$

Secondo la definizione di matrice unitaria:

$$U^\dagger U = U U^\dagger = \mathbf{I}_N \,. \tag{3.3}$$

Eguagliando la 3.1 con la 3.3 si può derivare:

$$U_r \cdots U_1 = U^\dagger \,, \tag{3.4}$$

quindi:

$$U = U_1^\dagger \cdots U_r^\dagger \,. \tag{3.5}$$

L'obiettivo del metodo di Li-Pelejo è quello di ottenere le matrici $U_1, ..., U_r$ al fine di derivare U secondo l'equazione 3.5.
Le matrici U_i utilizzate per la scomposizione sono matrici di tipo multi-controllo singolo-target: ovvero matrici in cui una rotazione generica T_i [1] è applicata ad un singolo qubit in funzione del valore dei restanti qubit.
Per esempio su con due qubit si possono definire quattro porte "controllate": (0T), (T0), (1T) e (T1) [2] e due porte "non controllate": (*T) e (T*). Le porte controllate applicano la generica rotazione T_i sul qubit target quando il qubit di controllo assume il valore richiesto dalla porta: 0 nel caso di porte (0T) o (T0), e 1 nel caso di porte (1T) o (T1). Invece le porte non controllate applicano la generica rotazione T_i indipendentemente dal valore assunto dall'altro qubit.

[1] In questo contesto si usa T_i per indicare una generica rotazione descritta da una generica matrice 2x2 formata da elementi T_{11}, T_{12}, T_{21}, T_{22}. Da non confondere con la porta T ovvero la porta di rotazione $R_z(\frac{\pi}{4})$.

[2] In questo contesto la notazione identifica il qubit più significativo a sinistra e qubit meno significativo a destra.

Data T_i una generica matrice 2×2:

$$T_i = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} . \tag{3.6}$$

Le porte controllate di tipo (1T) e (0T) sono rispettivamente:

$$(1T) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & T_{11} & T_{12} \\ 0 & 0 & T_{21} & T_{22} \end{bmatrix} , \tag{3.7}$$

$$(0T) = \begin{bmatrix} T_{11} & T_{12} & 0 & 0 \\ T_{21} & T_{22} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \tag{3.8}$$

Si può osservare come la moltiplicazione di ciascuna di queste matrici su un'altra matrice M modifichi solo le prime due colonne di M (nel caso della matrice 1T) e le ultime due colonne di M (nel caso della matrice 0T).
Le porte controllate di tipo (T1) e (T0) sono rispettivamente:

$$(T1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & T_{11} & 0 & T_{12} \\ 0 & 0 & 1 & 0 \\ 0 & T_{21} & 0 & T_{22} \end{bmatrix} , \tag{3.9}$$

$$(T0) = \begin{bmatrix} T_{11} & 0 & T_{12} & 0 \\ 0 & 1 & 0 & 0 \\ T_{21} & 0 & T_{22} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \tag{3.10}$$

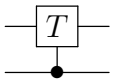
Si può osservare come la moltiplicazione di ciascuna di queste matrici con un'altra matrice M modifichi solo la seconda e la quarta colonna di M (nel caso della T1) e la prima e la terza colonna di M (nel caso della T0).

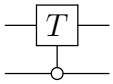
Le porte non controllate di tipo (*T) e (T*) sono rispettivamente:

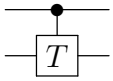
$$(*T) = \begin{bmatrix} T_{11} & T_{12} & 0 & 0 \\ T_{21} & T_{22} & 0 & 0 \\ 0 & 0 & T_{11} & T_{12} \\ 0 & 0 & T_{21} & T_{22} \end{bmatrix}, \tag{3.11}$$

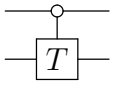
$$(T*) = \begin{bmatrix} T_{11} & 0 & T_{12} & 0 \\ 0 & T_{11} & 0 & T_{12} \\ T_{21} & 0 & T_{22} & 0 \\ 0 & T_{21} & 0 & T_{22} \end{bmatrix}. \tag{3.12}$$

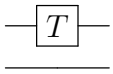
Si può osservare come la moltiplicazione di ciascuna di queste matrici su un altra matrice M modifichi tutte le colonne della matrice.
I circuiti quantistici risultanti sono [3]:

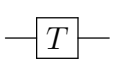
$(1T)$:  (3.13)

$(0T)$:  (3.14)

$(T1)$:  (3.15)

$(T0)$:  (3.16)

$(*T)$:  (3.17)

$(T*)$:  (3.18)

In sistemi con più di due qubit si possono individuare porte di tipo: "completamente controllate", "parzialmente controllate" e "non controllate". Nel caso di porte "completamente controllate", tutti i qubit escluso quello target sono di controllo. Le

[3] In questa tesi per i circuiti quantistici rappresentati viene usata la notazione di IBM dove il qubit più significativo è quello più in basso, mentre, il qubit meno significativo è quello più in alto.

porte sono "parzialmente controllate" quando almeno uno dei qubit, escluso quello target, non è controllo. Per esempio, una porta (0101T1) è una porta completamente controllata, mentre (01*1T1) è una porta parzialmente controllata. Invece, una porta del tipo (***T*) è una porta non controllata.

Dal punto di vista pratico, la scomposizione proposta da Li-Pelejo cerca di trasformare la matrice U nella matrice identità $\mathbf{I}_N$, eliminando uno dopo l'altro gli elementi non nulli fuori dalla diagonale, attraverso la moltiplicazione di U con le varie matrici U_i, da individuare sequenzialmente cercando di eliminare [4] ogni elemento triangolare inferiore di U. Nel caso di matrici unitarie l'eliminazione degli elementi della matrice triangolare inferiore garantisce l'eliminazione dei loro simmetrici sulla triangolare superiore [5].
L'ordine di eliminazione degli elementi della matrice triangolare inferiore e la matrice U_i sono calcolati in modo da non modificare gli elementi eliminati precedentemente.
Per la spiegazione teorica completa dell'algoritmo si rimanda all'articolo [9].

Per semplificare la comprensione dell'algoritmo si riporta un esempio.
La matrice unitaria di partenza è:

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}. \tag{3.19}$$

Il numero di qubit è: $n = 2$, per cui $N = 2^n = 4$.
Il numero massimo di porte richieste per la scomposizione è $r = \frac{N(N-1)}{2} = 6$.
L'ordine di scomposizione degli elementi della matrice triangolare inferiore è[6]:

- **colonna 0:** 1, 3, 2;

[4] In questo contesto per eliminare si intende di portare a 0 un determinato elemento della matrice.

[5] Questo dimostra l'equazione 3.2, in quanto gli elementi di una generica matrice triangolare inferiore diversi da 0 sono al più $\frac{N\cdot(N-1)}{2}$.

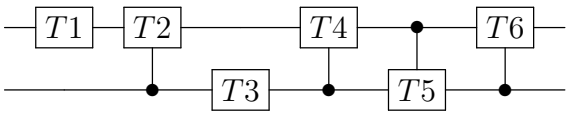
[6] In questa tesi, gli indici delle righe e delle colonne delle matrici vengono numerati da 0 a $2^n - 1$. Per cui nel caso di matrice 2×2 gli indici vanno da 0 a 3.

- **colonna 1:** 2, 3;
- **colonna 2:** 3;

Il tipo di porte U_i da utilizzare per l'eliminazione degli elementi della matrice triangolare inferiore sono rispettivamente:

- **colonna 0:** (*T), (1T), (T*);
- **colonna 1:** (1T), (T1);
- **colonna 2:** (1T);

Secondo questa notazione, gli elementi (1,0), (2,0) e (3,0) della matrice U sono eliminati con le matrici (*T), (1T) e (T*) rispettivamente. Discorso analogo per le altre colonne.
Lo schema di scomposizione risultante è:



(3.20)

Nel primo step della scomposizione viene puntato l'elemento (1,0). In questo caso l'elemento (1,0) della matrice U risulta già essere uguale a 0; per cui, la porta T_1 è la matrice identità $\mathbf{I}_2$. Essa può essere eliminata dal circuito in quanto il suo effetto è nullo.

Nel secondo step della scomposizione, viene puntato l'elemento (3,0); anche in questo caso il suo valore è già 0, per cui la porta T_2 risulta anche essa essere uguale alla matrice identità $\mathbf{I}_2$. Anche questa porta può essere eliminata dal circuito [7].

Nello terzo step della scomposizione viene puntato l'elemento (2,0), il suo valore è pari a $\frac{1}{\sqrt{2}}$. in questo caso per la creazione della matrice T_3 si prende in considerazione l'elemento puntato (di seguito denominato b) e l'elemento della stessa colonna presente sulla diagonale principale (di seguito denominato a, in questo caso uguale

[7] L'effetto di una porta identità controllata è nullo.

a $\frac{1}{\sqrt{2}}$); si calcola il modulo tra a e b come: $z = \sqrt{|a^2| + |b^2|}$, infine la matrice T_3 viene creata come:

$$T_3 = \frac{1}{z} \begin{bmatrix} \text{conj}(a) & \text{conj}(b) \\ -b & a \end{bmatrix}, \tag{3.21}$$

dove conj$(\cdot)$ è il complesso coniugato dell'elemento indicato tra le parentesi. Per cui la matrice T_3 risulta essere:

$$T_3 = \begin{bmatrix} \frac{1}{\sqrt{(2)}} & \frac{1}{\sqrt{(2)}} \\ -\frac{1}{\sqrt{(2)}} & \frac{1}{\sqrt{(2)}} \end{bmatrix}. \tag{3.22}$$

Di conseguenza considerando che la porta U_3 è una porta di tipo (T*) essa è uguale ad:

$$U_3 = \begin{bmatrix} \frac{1}{\sqrt{(2)}} & 0 & \frac{1}{\sqrt{(2)}} & 0 \\ 0 & \frac{1}{\sqrt{(2)}} & 0 & \frac{1}{\sqrt{(2)}} \\ -\frac{1}{\sqrt{(2)}} & 0 & \frac{1}{\sqrt{(2)}} & 0 \\ 0 & -\frac{1}{\sqrt{(2)}} & 0 & \frac{1}{\sqrt{(2)}} \end{bmatrix}. \tag{3.23}$$

Moltiplicando a destra la matrice di ingresso U per la U_3 si ottiene:

$$Y = U_3 \cdot U = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}. \tag{3.24}$$

Si può notare che l'elemento (2,0) sia stato correttamente portato a 0. Inoltre si può osservare come la matrice (T*), che come indicato precedentemente è una matrice che influisce su tutte le colonne della matrice originaria U, abbia portato a 0 anche l'elemento in posizione (3,1). Questo è vantaggioso perché consente di eliminare una porta dal circuito finale, essendo la porta per l'eliminazione dell'elemento (3,1) un'identità.

Nel quarto step della scomposizione viene puntato l'elemento (2,1). In questo caso l'elemento puntato risulta essere già uguale a 0; per cui la matrice T_4 è una matrice

identità, di conseguenza la porta U_4 può essere eliminata dal circuito finale.

Nel quinto step della scomposizione viene puntato l'elemento (3,1). Quest'elemento è stato eliminato precedentemente, per cui la matrice T_5 è l'identità $\mathbf{I}_2$ e la porta U_5 può essere eliminata dal circuito finale.

Nell'ultimo step della scomposizione viene puntato l'elemento in posizione (3,2). Anche quest'elemento risulta essere uguale a 0; ma in questo caso, essendo l'ultimo step di decomposizione, la matrice viene costruita nel seguente modo:

$$\begin{bmatrix} \text{conj}(U[2^{n-1}-1, 2^{n-1}-1]) & \text{conj}(U[2^n-1, 2^{n-1}-1]) \\ \text{conj}(U[2^{n-1}-1, 2^n-1]) & \text{conj}(U[2^n-1, 2^n-1]) \end{bmatrix}, \tag{3.25}$$

per cui la matrice T_6 non è la matrice identità, ma risulta essere la matrice "meno identità":

$$T_6 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.26}$$

Di conseguenza considerando che la porta U_6 è una porta di tipo (1T) essa è uguale a:

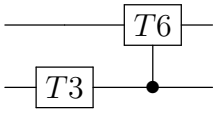
$$U_6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}. \tag{3.27}$$

Moltiplicando a destra la matrice Y per la U_6 si ottiene:

$$Y = U_6 \cdot Y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.28}$$

La scomposizione è corretta, in quanto la matrice residua è la matrice identità.
Il circuito finale risulta essere composto da sole due porte, un numero minore rispetto alle sei massime previste. È importante precisare che questa riduzione del numero di porte, dipende dai valori degli elementi della matrice iniziale U e non è garantita

per ogni matrice.



(3.29)

3.2 Implementazione

In questa sezione sono riportati i diagrammi di flusso adoperati per l'implementazione in Python del metodo di scomposizione di Li-Pelejo.
Per la realizzazione dei primi due passi base dell'algoritmo (mostrato in Figura 3.1) è stato considerato lo script Matlab realizzato da Li-Pelejo reperibile al link presente all'interno dell'articolo [9]. Tale script è stato tradotto dal linguaggio Matlab al linguaggio Python, eliminando bug ed aggiungendo parti di codice necessarie per renderlo compatibile con gli algoritmi successivi, fondamentali per completare la scomposizione della matrice unitaria ed ottenere il circuito in OpenQASM 2.0.

3.2.1 Diagramma di flusso principale

Nella Figura 3.1 è riportato il diagramma di flusso principale della scomposizione di Li-Pelejo.
Il primo passo di scomposizione, prevede il calcolo dell'ordine di eliminazione degli elementi della matrice unitaria di ingresso. Insieme all'ordine di eliminazione viene anche creato lo schematico base per la scomposizione della matrice, dove sono indicate solamente il tipo di porte U_i (per esempio su 4 qubit una possibile porta potrebbe essere (*11T)), e non i valori degli elementi delle singole matrici T_i. Tali valori vengono calcolati nel secondo blocco del diagramma di flusso e salvati in una lista di liste V. Contestualmente al calcolo dei valori delle matrici T_i sono effettuati dei test per verificare, che ogni matrice U_i prodotta, sia effettivamente una matrice unitaria. Nel caso in cui almeno una matrice U_i non sia unitaria, viene segnalato l'errore, indicando il numero i della matrice non unitaria. In caso di assenza di errori, viene eseguita una semplificazione dello schematico A e della matrice V; in questa fase di semplificazione vengono eliminate tutte le righe di V che contengono la lista [1,0,0,1]; tale lista corrisponde alla matrice identità, che come visto nella teoria può

essere eliminata in quanto il suo effetto è nullo. Contestualmente vengono eliminate anche le relative linee dello schematico A (un esempio e mostrato nella Figura 3.2).

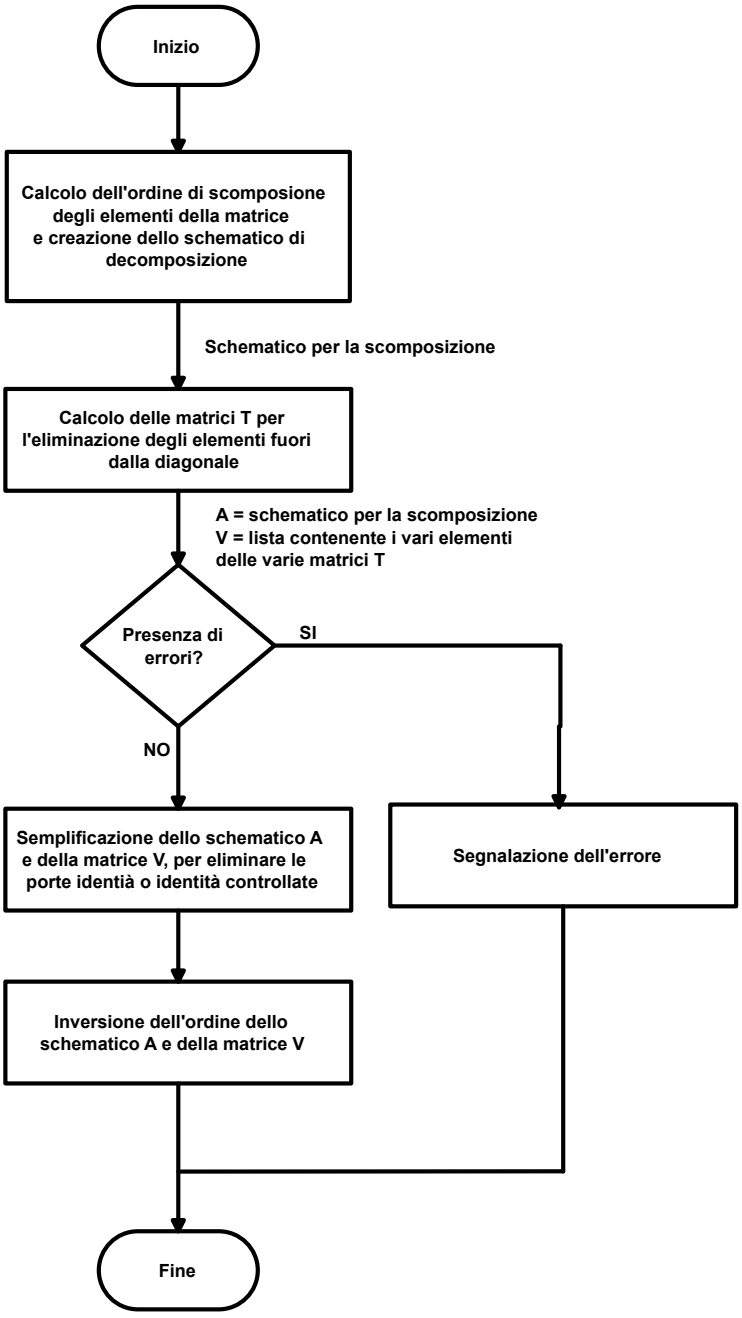


Figura 3.1: Diagramma di flusso per l'implementazione dell'algoritmo di scomposizione di Li-Pelejo.

L'uscita dell'algoritmo sono due liste di liste: una chiamata A che contiene lo schematico di scomposizione, mentre l'altra chiamata V contiene gli elementi delle varie matrici T_i che compongono lo schematico. Un esempio di A e V è visibile nella Figura 3.2. È importante osservare che le gli elementi di A e V sono collegati dalla loro posizione, ovvero che gli elementi della matrice T_1, appartenente alla prima linea di A, si trovano nella prima linea di V; quelli della seconda matrice T_2, appartenenti alla seconda linea di A, si trovano nella seconda linea di V, e cosi via.
In uscita dall'algoritmo di scomposizione di Li-Pelejo implementato, A e V hanno la stessa lunghezza. L'ultimo passo dell'algoritmo di Li-Pelejo, prevede l'inversione dell'ordine delle liste che compongono lo schematico A e la lista di liste V, in modo da passare dalle matrici $U_r \cdots U_1$ alle matrici $U_1^\dagger \cdots U_r^\dagger$ che descrivono la matrice di ingresso U.

```
A = [ ['T', '*', '1'],
      ['*', 'T', '*'],
      ['T', '1', '*'],
      ['T', '*', '*'] ]
```

```
V =  [ [(0.9-0.1j), (-0.3+0.08j), (0.3+0.008j), (0.9+0.1j)],
       [(0.6+0j), (0.7-0j), (-0.7-0j), (0.6-0j)],
       [(0.7+0.4j), (0.4-0.2j), (-0.4-0.2j), (0.7-0.4j)],
       [(0.4-0.5j), (0.06-0.7j), (-0.06-0.7j), (0.4+0.5j)] ]
```

Figura 3.2: Esempio generico di A e V in uscita dall'algoritmo di scomposizione di Li-Pelejo implementato.

3.2.2 Calcolo dell'ordine di eliminazione e creazione dello schematico

Il primo step dell'algoritmo di scomposizione prevede il calcolo dell'ordine di scomposizione e la generazione dello schematico base. Il diagramma di flusso di questo blocco è mostrato in Figura 3.3.

Figura 3.3: Diagramma di flusso del blocco "calcolo dell'ordine di scomposizione e creazione dello schematico" dell'algoritmo di Li-Pelejo implementato.

Il primo passo consiste nel calcolare l'ordine di scomposizione delle colonne. Si ricorda che si procede in ordine, eliminando prima tutti gli elementi della matrice triangolare inferiore della prima colonna, poi tutti quelli della seconda e così via. Gli indici della colonna di ogni elemento da eliminare sono contenuti in una lista y. Il secondo passo prevede il calcolo dell'ordine delle righe per l'eliminazione degli elementi della matrice triangolare inferiore. Al contrario di quanto accade per le colonne, per le righe non si procede in ordine, bensì, l'ordine è condizionato dal

fatto che non bisogna modificare gli elementi precedentemente eliminati. Una lista x contiene l'indice di riga di ogni elemento da eliminare.
Il terzo passo, prevede la creazione dello schematico A usato per la scomposizione della matrice di ingresso U. Il calcolo dell'ordine delle righe e la creazione dello schematico viaggiano di pari passo, in quanto, in questa fase dell'algoritmo di scomposizione di Li-Pelejo si cerca di estendere la struttura per $n-1$ qubit su n qubit, seguendo le regole riportate nella teoria illustrata nell'articolo [9].
Un esempio su 2 qubit di y, x e A è riportato in Figura 3.4.

```
y = [0, 0, 0, 1, 1, 2]

x = [1, 3, 2, 2, 3, 3]

A = [ ['*', 'T'],
      ['1', 'T'],
      ['T', '*'],
      ['1', 'T'],
      ['T', '1'],
      ['1', 'T'] ]
```

Figura 3.4: Esempio di y, x e A in uscita dal primo blocco di scomposizione dell'algoritmo di Li-Pelejo.

Le liste x e y e lo schematico A sono fondamentali per il calcolo delle matrici T_i nel secondo blocco dell'algoritmo di scomposizione. È importante sottolineare come sia x che y che A non dipendano dai valori degli elementi della matrice di ingresso U, ma solamente dal numero di qubit su cui essa opera.

3.2.3 Calcolo delle matrici T_i per l'eliminazione degli elementi fuori dalla diagonale.

Nel secondo step dell'algoritmo vengono calcolati i valori delle matrici T_i per l'eliminazione degli elementi fuori dalla diagonale della matrice di ingresso U. Il diagramma di flusso del secondo blocco della scomposizione è raffigurato in Figura 3.5.

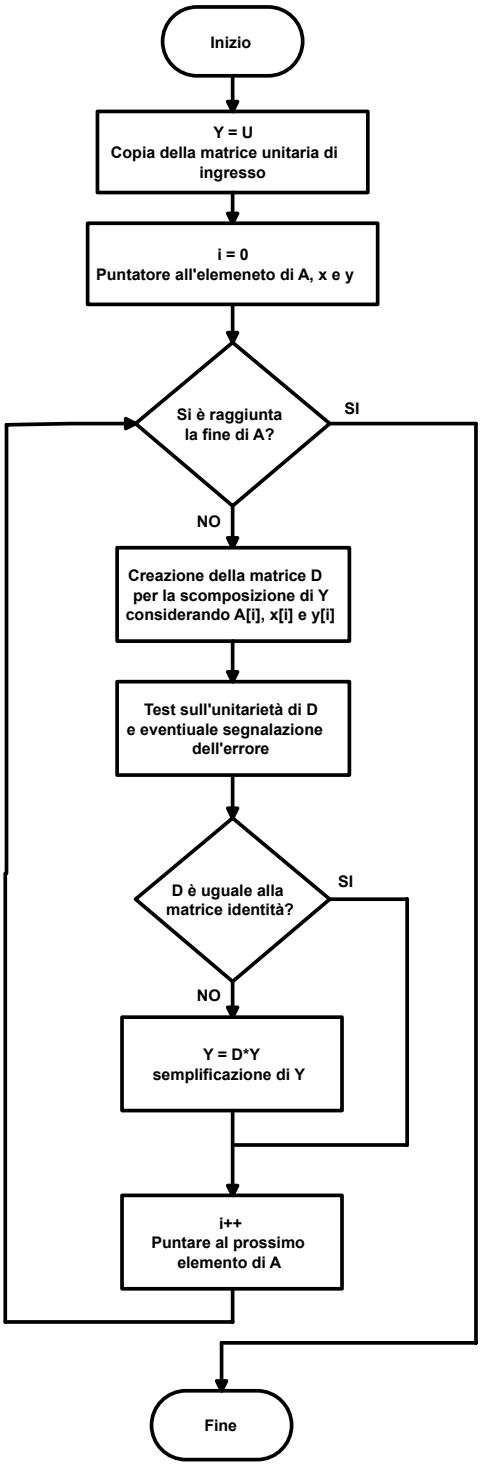


Figura 3.5: Diagramma di flusso per l'implementazione del blocco di "Calcolo delle matrici T_i per l'eliminazione degli elementi fuori dalla diagonale" dell'algoritmo di scomposizione di Li-Pelejo.

In questo step sono calcolati gli elementi delle varie matrici T_i, prendendo in considerazione il tipo di matrice riportato nello schematico A e l'elemento della matrice Y[8], individuato tramite i valori delle liste x e y.
In questa fase c'è la possibilità di abilitare i test sulle matrici generate per la scomposizione della matrice di ingresso U, ponendo a '1' il relativo flag `testEN`. La tolleranza numerica con la quale vengono eseguiti i test sulle varie matrici viene fornita alla funzione tramite la variabile `toll`. Alla fine della scomposizione si testa anche che il residuo sia effettivamente la matrice identità.
La verifica funzionale dell'algoritmo è stata accuratamente testata, ma è stata comunque lasciata la possibilità di abilitare i test nell'implementazione finale, nel caso in cui si voglia ottenere una scomposizione con un errore numerico al di sotto di una certa soglia. È importante sottolineare che l'abilitazione di questi test impatta sulla latenza di scomposizione della matrice come illustrato nella sezione successiva.

3.3 Semplificazione dello schematico A e di V

In questo blocco del diagramma di flusso della scomposizione di Li-Pelejo, viene semplificato lo schematico A e la lista di liste V, per eliminare tutte le linee contenenti matrici identità o matrici identità controllate.
Nella Figura 3.6 viene riportato il diagramma di flusso della semplificazione di A e V.
Questa operazione viene eseguita in modo da ridurre la quantità di memoria utilizzata dall'algoritmo. Inoltre, essa è eseguita prima dell'operazione di inversione di A e V, al fine di ridurre la latenza dell'operazione di inversione che agirà su liste con un numero inferiore di elementi.

[8]La matrice Y è la matrice copia della matrice di ingresso U, a cui vengono moltiplicate le varie matrici U_i per eliminare, ad ogni iterazione, gli elementi fuori dalla diagonale.

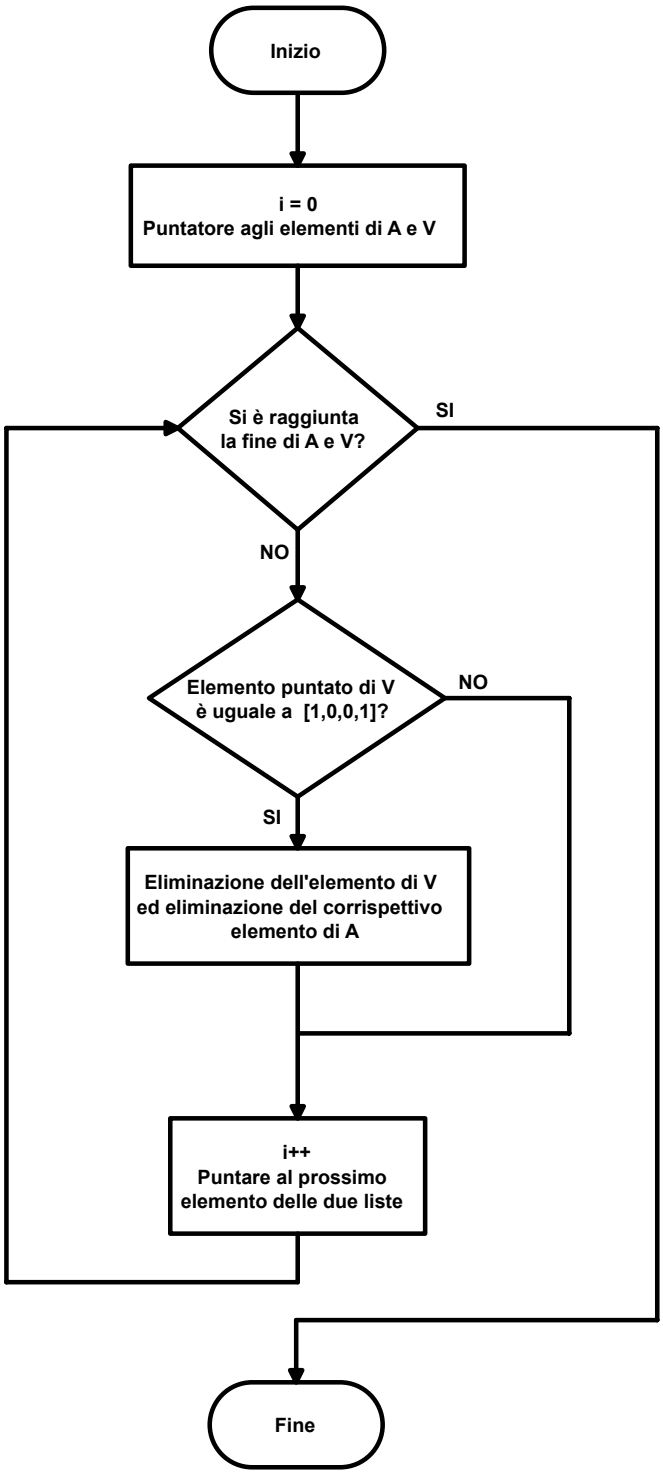


Figura 3.6: Diagramma di flusso per l'implementazione del blocco di "Semplificazione dello schematico A e della lista di liste V" dell'algoritmo di scomposizione di Li-Pelejo.

3.3.1 Inversione dell'ordine di A e V

L'ultimo step del diagramma di flusso della scomposizione di Li-Pelejo prevede l'inversione dello schematico A e della lista V. In Figura 3.9 Viene riportato il diagramma di flusso di questo blocco. L'operazione di "inversione" è necessaria perché l'algoritmo di scomposizione di Li-Pelejo fornisce in uscita:

$$U_r \cdots U_1 = U^\dagger \,, \tag{3.30}$$

mentre ciò che è necessario ottenere è:

$$U = U_1^\dagger \cdots U_r^\dagger \tag{3.31}$$

Per effettuare questa "inversione" sia lo schematico A che la lista di liste V vengono invertite, in modo che l'ultima linea di A e V diventi la prima linea di A e V rispettivamente. In oltre per effettuare l'operazione di *dagger* (che corrisponde alle operazioni di trasposizione della matrice più quella di complesso coniugato) viene eseguita l'operazione di complesso coniugato su ogni lista di V ed invertiti i due elementi centrali per effettuare la trasposizione. Lo schematico A viene invertito anche "orizzontalmente", questa operazione viene fatta per rispettare la convenzione di rappresentazione di Qiskit che prevede di avere il most significand qubit nell'ultima colonna di destra. Alla fine dell'inversione la prima porta sarà la prima presente nella lista dello schematico A, ed il qubit più significativo sarà quello più a destra e quello meno significativo sarà quello più a sinistra.
Per chiarire le operazioni eseguite in questo blocco, viene riportato un esempio:
ipotizzando di avere all'ingresso del blocco di inversione di A e V le liste riportate in Figura 3.7:

```
A = [ ['*', '*', 'T'],
      ['*', '1', 'T'],
      ['*', 'T', '*'],
      ['1', '*', 'T'] ]

V =  [ [(0.4-0.5j), (0.06-0.7j), (-0.06-0.7j), (0.4+0.5j)],
       [(0.7+0.4j), (0.4-0.2j), (-0.4-0.2j), (0.7-0.4j)],
       [(0.6+0j), (0.7-0j), (-0.7-0j), (0.6-0j)],
       [(0.9-0.1j), (-0.3+0.08j), (0.3+0.008j), (0.9+0.1j)] ]
```

Figura 3.7: Esempio di liste A e V prima dell'operazione di inversione.

alla fine del blocco di inversione si otterranno le liste riportate in Figura 3.8:

```
A = [ ['T', '*', '1'],          V =  [ [(0.9-0.1j), (-0.3+0.08j), (0.3+0.008j), (0.9+0.1j)],
      ['*', 'T', '*'],                 [(0.6+0j), (0.7-0j), (-0.7-0j), (0.6-0j)],
      ['T', '1', '*'],                 [(0.7+0.4j), (0.4-0.2j), (-0.4-0.2j), (0.7-0.4j)],
      ['T', '*', '*'] ]                [(0.4-0.5j), (0.06-0.7j), (-0.06-0.7j), (0.4+0.5j)] ]
```

Figura 3.8: Esempio di liste A e V dopo dell'operazione di inversione.

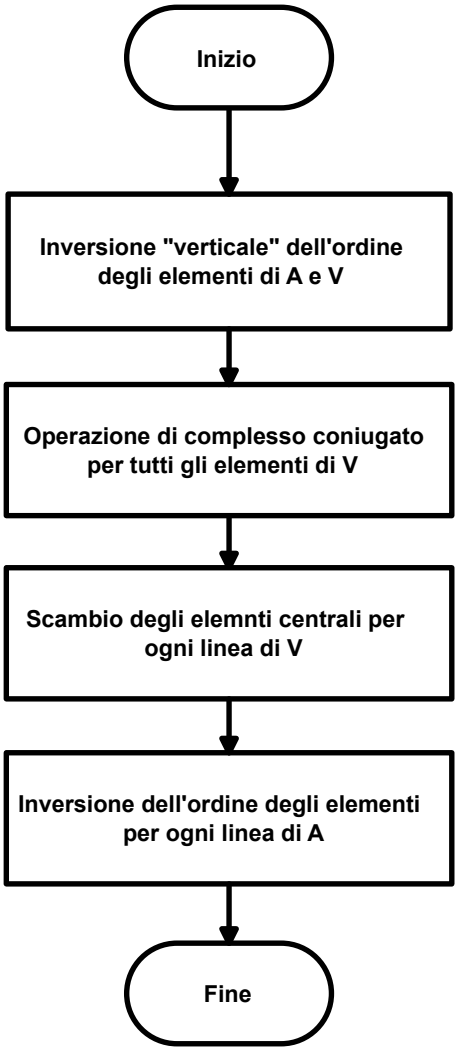


Figura 3.9: Diagramma di flusso per l'implementazione del blocco di "inversione dell'ordine di a e V" dell'algoritmo di scomposizione di Li-Pelejo.

3.4 Risultati

In questa sezione vengono riportati e discussi i risultati ottenuti per la scomposizione con Li-Pelejo di varie matrici unitarie, in termini di tempo impiegato per la scomposizione ed errore numerico.

3.4.1 Tempo di scomposizione

Nel calcolo del tempo di scomposizione, le matrici unitarie utilizzate sono matrici unitarie formate da elementi casuali. Questo implica che la parte reale e immaginaria di ogni elemento della matrice sono diverse da zero, per cui i risultati ottenuti in questa sezione sono da considerarsi "worst-case" per quanto concerne il tempo di scomposizione.

Nella Figura 3.10 è riportato il grafico per il tempo di scomposizione dell'algoritmo di Li-Pelejo al crescere del numero di qubit della matrice unitaria di ingresso U. Nella Figura 3.11 sono riportati gli stessi dati ma in scala semi-logaritmica.
I valori riportati nei grafici sono relativi alla media di dieci scomposizioni per ogni valore del numero di qubit.
Nelle due figure sono messi a confronto i tempi con e senza l'abilitazione dei test (`testEN:1` e `testEN:0` rispettivamente) sulle varie matrici prodotte dall'algoritmo per la scomposizione della matrice di ingresso.
È possibile notare che all'aumentare del numero di qubit si ha un aumento del tempo di scomposizione di entrambe le curve, concorde con l'aumento esponenziale delle matrici e anche con l'aumento esponenziale del numero di matrici usate per la scomposizione.
Inoltre, è possibile osservare che al crescere del numero di qubit cresce la distanza tra la curva con abilitazione dei test e la curva senza l'abilitazione dei test. Questo fenomeno è ancora una volta dovuto all'aumento esponenziale delle dimensioni e al numero delle matrici coinvolte. Infatti, per verificare che ogni matrice generata dall'algoritmo sia unitaria, si moltiplica tale matrice per la sua trasposta e complessa coniugata, e si verifica che il risultato di questa moltiplicazione sia uguale alla matrice identità considerando una tolleranza impostata dall'utente.

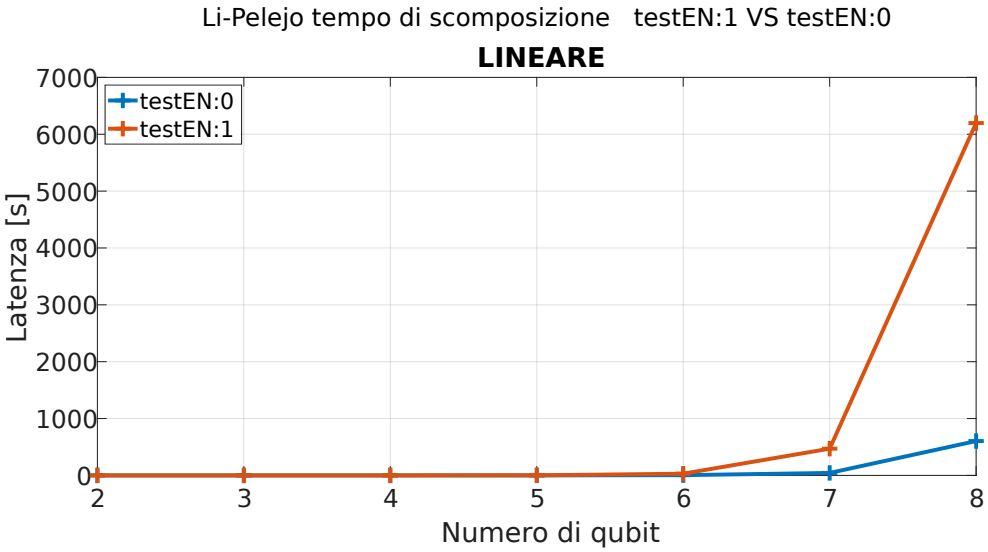


Figura 3.10: Tempo di scomposizione secondo l'algoritmo di Li-Pelejo di matrici unitarie casuali, in scala lineare.

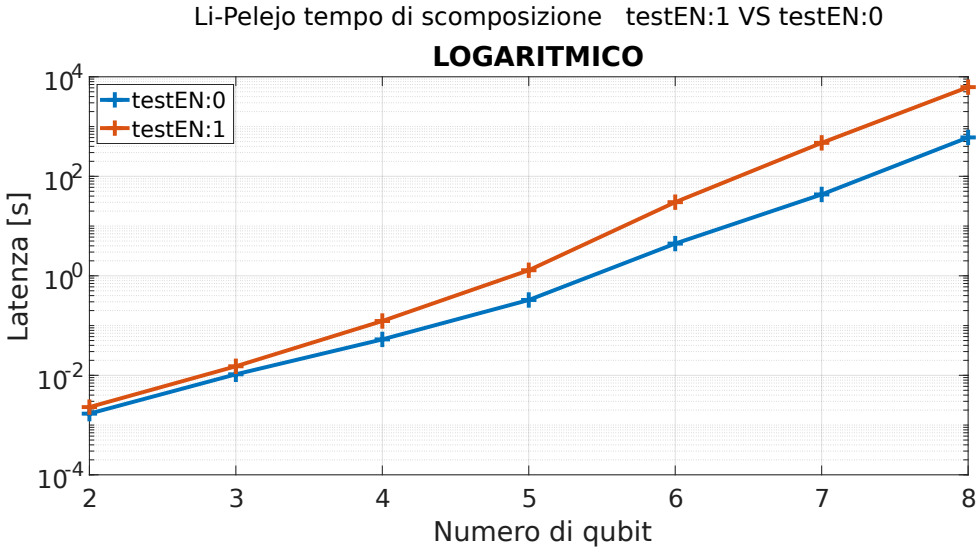


Figura 3.11: Tempo di scomposizione secondo l'algoritmo di Li-Pelejo di matrici unitarie casuali, in scala semi-logaritmica.

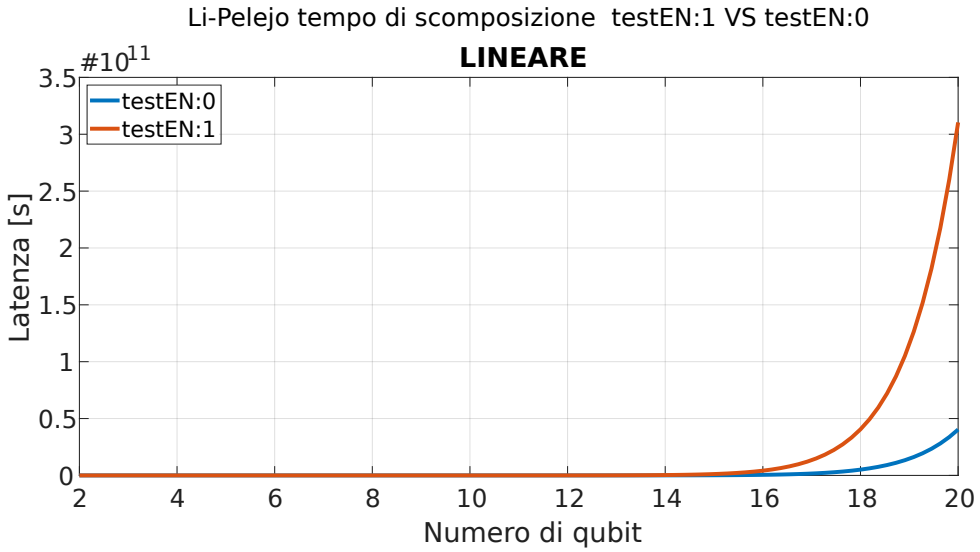


Figura 3.12: Interpolazione del tempo di scomposizione secondo l'algoritmo di Li-Pelejo di matrici unitarie casuali, in scala lineare.

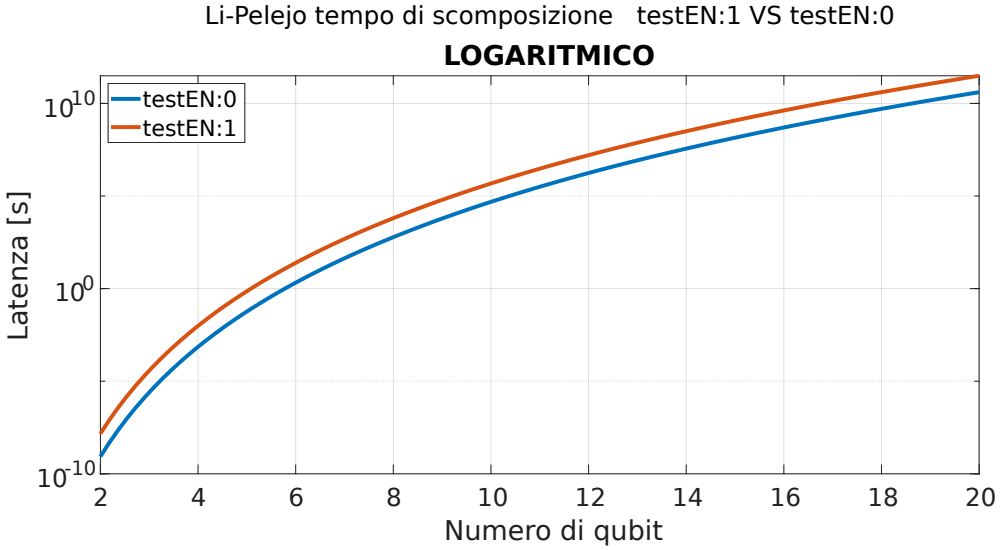


Figura 3.13: Interpolazione del tempo di scomposizione secondo l'algoritmo di Li-Pelejo di matrici unitarie casuali, in scala semi-logaritmica.

Nelle Figure 3.12 e 3.13 sono riportati i grafici relativi all'interpolazione dei dati. Tale interpolazione è stata eseguita usando un toolbox di Matlab per estrapolare la funzione interpolante, ed estendendo il range da due qubit a venti qubit. Questi due grafici sono da interpretare come una previsione del tempo di scomposizione di matrici unitarie coinvolgenti più di otto qubit.

3.4.2 Errore numerico

In questa sezione si riporta l'errore numerico riscontrato durante la scomposizione di matrici unitarie casuali al crescere del numero di qubit. Per errore numerico si intende la distanza che intercorre tra la matrice costruita come prodotto tra la matrice da testare e la sua trasposta coniugata, e la matrice identità. Sono stati eseguiti dieci simulazioni per ogni valore del numero di qubit, ed i valori riportati nei grafici riguardano gli errori più elevati riscontrati in valore assoluto.

Nella Figura 3.14 è riportato il grafico dell'errore numerico generato durante la scomposizione di Li-Pelejo. Tale errore cresce al crescere del numero di qubit; questo effetto può essere spiegato con l'aumento del numero di matrici usate per la scomposizione in quanto vengono moltiplicate più matrici tra loro, operazioni comportanti una crescita dell'errore.
Nella Figura 3.15 viene riportato l'interpolazione dell'errore numerico, estendendo il range da due a venti qubit. Questo grafico può essere visto come una previsione dell'errore numerico per un numero di qubit maggiore di otto.
Si può notare come l'errore rimanga sotto $8 \cdot 10^{-13}$ anche lavorando con venti qubit.

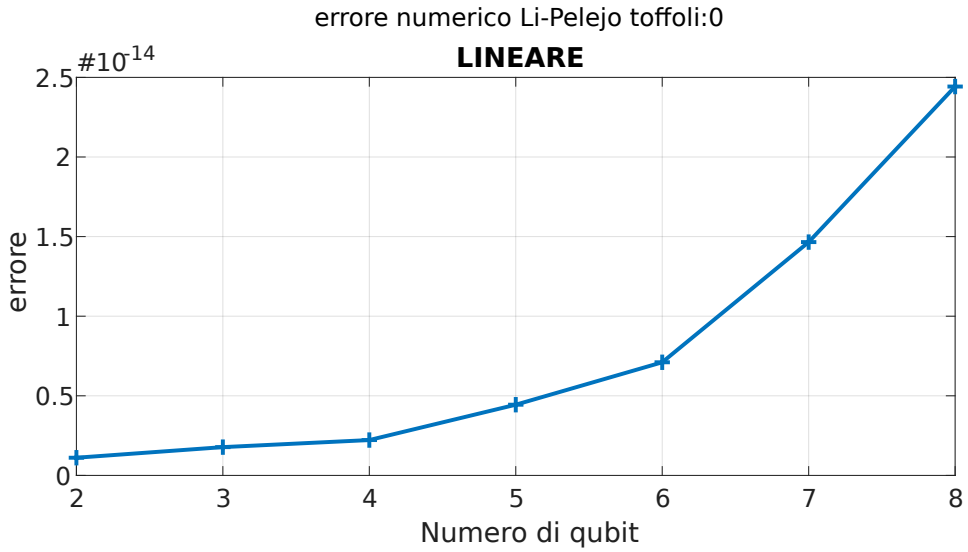


Figura 3.14: Errore numerico riscontrato nelle simulazioni dell'algoritmo di scomposizione proposto da Li-Pelejo.

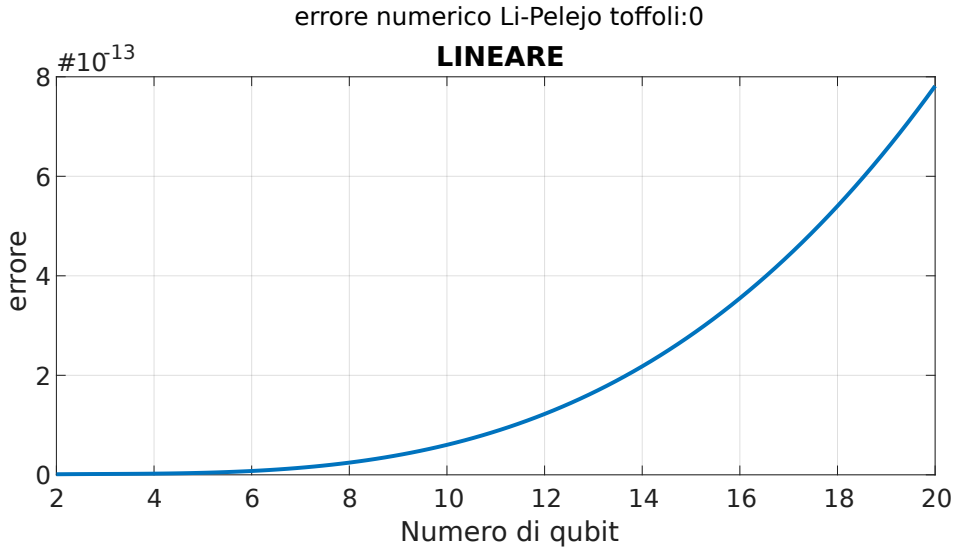


Figura 3.15: Interpolazione dell'errore numerico riscontrato nelle simulazioni dell'algoritmo di scomposizione proposto da Li-Pelejo.

3.4.3 Tabella riassuntiva

Nella tabella 3.1 vengono riportati i valori: del tempo di scomposizione con test abilitati/disabilitati e l'errore numerico generato nella scomposizione utilizzando l'algoritmo di Li-Pelejo.

Numero di qubit	2	3	4	5	6	7	8
Tempo di scomposizione Li-Pelejo testEN:0 [s]	0.0017	0.0104	0.0527	0.3265	4.4403	43.191	604.30
Tempo di scomposizione Li-Plejo testEN:1 [s]	0.0023	0.0152	0.1234	1.2933	30.200	470.02	6197.4
Errore numerico Li-Pelejo	1.1e-15	1.7e-15	2.2e-15	4.4e-15	7.1e-15	1.5e-14	2.4e-14

Tabella 3.1: Tabella riassuntiva con i valore delle scomposizione di matrici unitarie tramite l'algoritmo di Li-Pelejo.

Comparando i vari risultati ottenuti di tempo di scomposizione si nota come l'abilitazione dei test, sull'unitarietà delle varie matrici prodotte, aumenti notevolmente il tempo richiesto per la scomposizione. Considerando che la crescita dell'errore numerico non è così accentuata da diventare critica in circuiti quantistici comuni, si suggerisce di disabilitare i test in modo da ridurre il tempo richiesto per la scomposizione, e di abilitarli, solo in caso in cui si voglia essere certi di ottenere una scomposizione con un errore numerico al di sotto di una certa soglia.

Capitolo 4

Scomposizione di Nielsen-Chuang e recupero di fase

In questo capitolo è descritto l'algoritmo di scomposizione ricorsivo proposto da Michael A. Nielsen e Isaac L. Chuang di seguito chiamato algoritmo di Nielsen-Chuang, usato per scomporre le porte di tipo multi-controllo singolo-target in porte di tipo singolo-controllo singolo-target. Inoltre, viene spiegato il meccanismo chiamato *recupero di fase*, che permette di scrivere le porte controllate secondo lo standard dell'OpenQASM 2.0 (Tipo $CU1$ e $CU3$).
La teoria alla base dell'algoritmo di Nielsen-Chuang è dettagliata in [2].

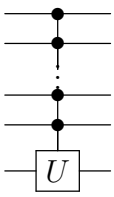
4.1 Teoria

In questa sezione viene spiegata la teoria alla base dei due algoritmi: la scomposizione ricorsiva di Nielsen-Chuang e la procedura del recupero di fase.

4.1.1 Scomposizione ricorsiva di Nielsen-Chuang

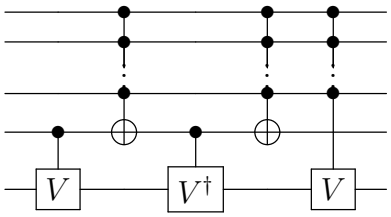
Questa scomposizione ricorsiva, permette di scomporre porte multi-controllo singolo-target in porte singolo-controllo singolo-target. Si tratta di una scomposizione *ricorsiva*, in quanto ad ogni iterazione si riduce di uno il numero di controlli di una data porta, per cui nel caso di più di due controlli è necessario applicare più volte il metodo di scomposizione sulla stessa porta.

Supponendo di avere in ingresso all'algoritmo la seguente generica porta multi-controllo singolo-target:



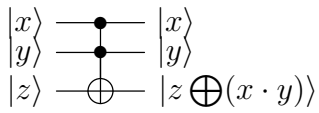
(4.1)

con n qubit di controllo, il circuito scomposto risulta essere:



(4.2)

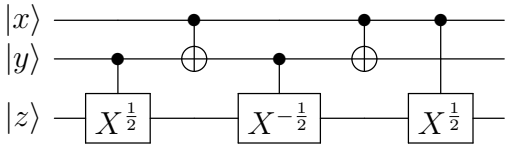
Dove la matrice V è legata alla matrice U dalla relazione: $V \cdot V = U$.
Si può notare come le due porte $C^{n-1}X$ e l'ultima porta V siano porte controllate su *n-1* qubit di controllo. Pertanto esse dovranno essere scomposte ulteriormente dall'algoritmo, per arrivare ad avere un numero di qubit di controllo pari ad uno.
Per dimostrare che il circuito scomposto è equivalente all'originale, viene analizzata la scomposizione della porta Toffoli.



(4.3)

La porta Toffoli, come riportato nel capitolo 1, cambia il valore del qubit z se e solo se tanto il valore del qubit x quanto il valore del qubit y sono uguali ad $|1\rangle$.

Il circuito scomposto della porta Toffoli risulta essere:



(4.4)

Si analizza l'evoluzione di $|z\rangle$ nel circuito scomposto nelle varie combinazioni di $|x\rangle$ e $|y\rangle$:

- Ipotizzando che sia $|x\rangle$ che $|y\rangle$ siano uguali a $|0\rangle$, nessuna rotazione viene eseguita su $|z\rangle$.
- Nel caso in cui $|x\rangle$ sia $|1\rangle$ e $|y\rangle$ sia $|0\rangle$: su $|z\rangle$ vengono applicate le rotazioni $X^{-\frac{1}{2}}$ e $X^{\frac{1}{2}}$ ottenendo $X^{-\frac{1}{2}} \cdot X^{\frac{1}{2}} \cdot |z\rangle = |z\rangle$.
- Nel caso in cui $|x\rangle$ sia $|0\rangle$ e $|y\rangle$ sia $|1\rangle$: su $|z\rangle$ vengono applicate le rotazioni $X^{\frac{1}{2}}$ e $X^{-\frac{1}{2}}$ ottenendo $X^{\frac{1}{2}} \cdot X^{-\frac{1}{2}} \cdot |z\rangle = |z\rangle$.
- Nel caso in cui $|x\rangle$ sia $|1\rangle$ e $|y\rangle$ sia $|1\rangle$: su $|z\rangle$ vengono applicate le rotazioni $X^{\frac{1}{2}}$ e $X^{\frac{1}{2}}$ ottenendo $X^{\frac{1}{2}} \cdot X^{\frac{1}{2}} \cdot |z\rangle = X \cdot |z\rangle$ che corrisponde a cambiare il valore di $|z\rangle$.

Per cui il circuito scomposto si comporta esattamente come la porta Toffoli originaria.

4.1.2 Recupero di fase

In questo paragrafo viene spiegata la teoria che sta alla base della procedura del recupero di fase. Questa procedura viene utilizzata per scrivere le porte controllate ottenute ricorsivamente secondo lo standard dell'OpenQASM 2.0, caratterizzato dall'uso di porte $CU1$ e $CU3$.
Nell'OpenQASM 2.0, le porte $U3$ vengono descritte in termini di θ, ϕ e λ ricavati secondo la seguente relazione:

$$U3(\theta, \phi, \lambda) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{j\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{j\phi}\sin\left(\frac{\theta}{2}\right) & e^{j(\lambda+\phi)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix}. \tag{4.5}$$

Allo stesso modo le porte di tipo $U1$ sono descritte in termini di λ, ricavato dalla seguente relazione:

$$U1(\lambda) = \begin{bmatrix} 1 & 0 \\ 0 & e^{j\lambda} \end{bmatrix}. \tag{4.6}$$

Si può notare che sia nel caso di porta $U3$ che nel caso di porta $U1$ il termine nella posizione (0,0) è un termine reale (1 nel caso di porta $U1$ e $\cos\left(\frac{\theta}{2}\right)$ nel caso di porta $U3$).
Ipotizzando di avere in uscita dall'algoritmo di Nielsen-Chuang una porta descritta dalla seguente matrice complessa V:

$$V = \begin{bmatrix} (0.5+0.5j) & (0.5-0.5j) \\ (0.5-0.5j) & (0.5+0.5j) \end{bmatrix}. \tag{4.7}$$

La prima operazione da eseguire per poter descrivere le porte nello standard del OpenQASM 2.0 è quella di eliminare la fase dell'elemento in posizione (0,0) della matrice. In questo caso la fase dell'elemento (0,0) risulta essere di:

$$fase = \arctan\left(\frac{0.5}{0.5}\right) \cong 0.785\,. \tag{4.8}$$

Per cui normalizzando l'intera matrice V diventa:

$$V_{norm} = \frac{V}{e^{j\cdot fase}} = \begin{bmatrix} 0.7071 & -0.7071j \\ -0.7071j & 0.7071 \end{bmatrix}. \tag{4.9}$$

Si può notare che la matrice V_{norm} è una matrice di tipo $U3$, in quanto gli elementi sull'anti-diagonale sono diversi da zero; per cui si possono estrarre i valori di θ, ϕ e λ da utilizzare nella descrizione in OpenQASM 2.0.
Dall'elemento in posizione (0,0) si può ricavare il valore di θ:

$$\theta = 2 \cdot \arccos(0.7071) \cong 1.57\,. \tag{4.10}$$

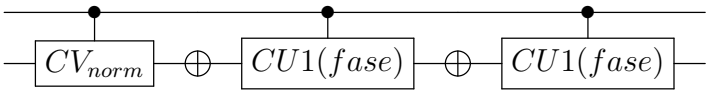
Conoscendo il valore di θ dagli elementi in posizione (0,1) e (1,0) si possono ricavare rispettivamente λ e ϕ:

$$e^{j\cdot\lambda} = \frac{-0.7071j}{-\sin\left(\frac{\theta}{2}\right)} \cong 1j \longrightarrow \lambda = 0\,, \tag{4.11}$$

$$e^{j\cdot\phi} = \frac{-0.7071j}{\sin\left(\frac{\theta}{2}\right)} \cong -1j \longrightarrow \phi = 0\,. \tag{4.12}$$

Nel caso di porte controllate la porta CV e CV_{norm} non sono equivalenti.
Per rendere equivalente la porta CV_{norm} alla porta CV è necessario aggiungere altre

porte, come mostrato di seguito:



(4.13)

Il parametro λ delle due $CU1$ chiamato *fase* è proprio la fase di $\mathbf{V}$[i][0] (nell'esempio riportato sopra la *fase* è uguale a 0.785).
Nel caso di porta controllata di tipo *U1* la procedura sul recupero di fase è la stessa riportata nel esempio.

4.2 Implementazione

In questa sezione sono riportati i diagrammi di flusso per l'implementazione in Python dell'algoritmo di scomposizione di Nielsen-Chuang e del recupero di fase.

4.2.1 Implementazione dell'algoritmo di Nielsen-Chuang

In questa sotto-sezione è spiegata l'implementazione dell'algoritmo di Nielsen-Chuang, mostrando i diagrammi di flusso con gli step principali dell'algoritmo.

Nella Figura 4.1 è riportato il diagramma di flusso più ad alto livello dell'implementazione dell'algoritmo di Nielsen-Chuang. Questo algoritmo è composto da due step: il primo consiste nel trasformare tutti i controlli di tipo '0' in controlli di tipo '1', il secondo step, invece, è la vera e propria implementazione dell'algoritmo ricorsivo.

Nella Figura 4.2 è riportato il diagramma di flusso per la trasformazione dei controlli di tipo '0' in controlli di tipo '1'. Questa operazione è necessaria perché la scomposizione di Nielsen-Chuang prevede che tutti i controlli siano di tipo '1'[1]. Per effettuare la conversione da controlli di tipo '0' a controlli di tipo '1', è necessario anteporre, nella linea dello schematico prima del controllo di tipo '0', una porta X, sostituire il

[1]Per controlli di tipo '1' si intendono qubit di controllo in cui la rotazione sul qubit target viene effettuata se il valore del qubit di controllo è uguale a 1. Viceversa, per controlli di tipo '0' si intendono qubit di controllo in cui la rotazione sul qubit target viene effettuata se il valore del qubit di controllo è uguale a 0.

controllo '0' con un controllo '1' ed post-porre nella linea successiva un'altra porta X. Nella Figura 4.3 è riportato un esempio. E' importante sottolineare come questa trasformazione dei controlli non modifichi la lista di lista V. Per risparmiare memoria, è stato deciso di non salvare il valore delle porte X introdotte da qui in avanti; tali porte verranno indicate con una 'X' direttamente all'interno dello schematico. Questo implica che le due liste di liste, lo schematico A e la lista di liste V, non avranno più la stessa lunghezza, ma saranno sempre collegati in base all'ordine degli elementi, quindi gli elementi della prima matrice di tipo T_i saranno contenuti nella prima linea di V, gli elementi della seconda matrice di tipo T_i saranno contenuti nella seconda linea di V, e cosi via.

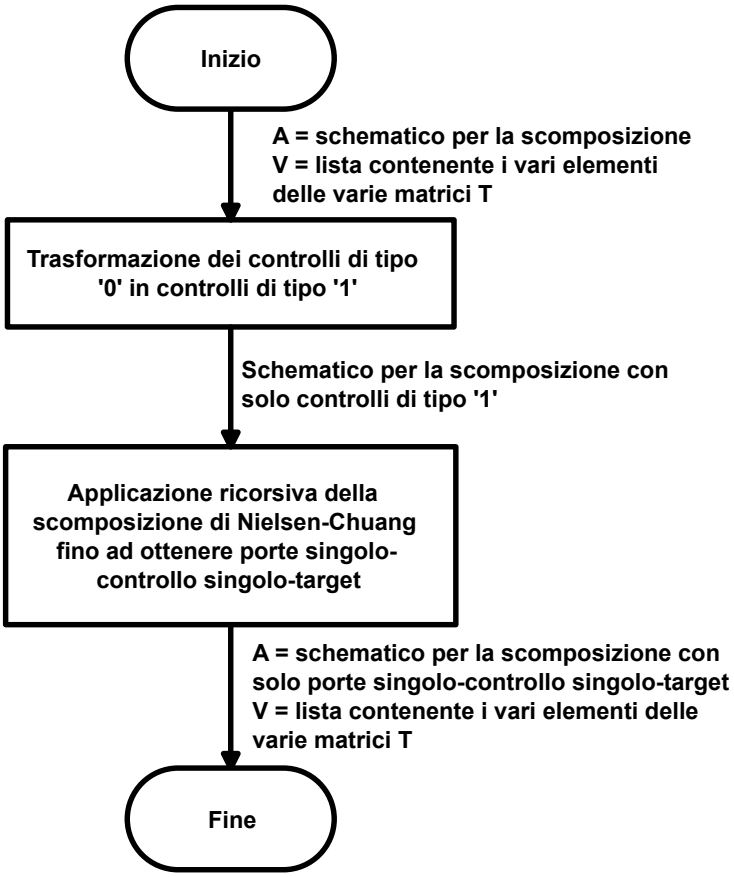


Figura 4.1: Diagramma di flusso principale dell'implementazione dell'algoritmo di Nielsen-Chuang.

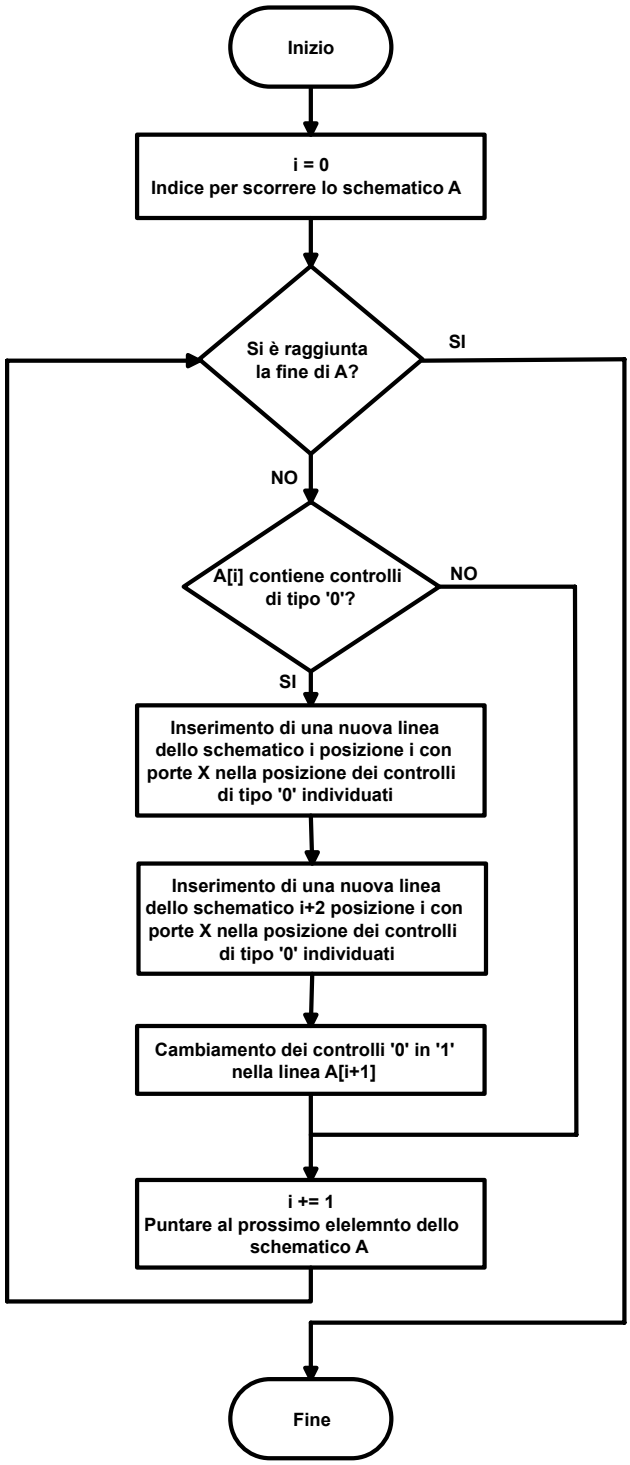


Figura 4.2: Diagramma di flusso dell'algoritmo per la trasformazione di controlli di tipo '0' in controlli di tipo '1'.

Prima della conversione dei controlli di tipo '0':

A = [['*', '1', '1', 'T'],
['T', '0', '*', '1']]

V = [[(0.5+0.5j), (0.5-0.5j), (0.5-0.5j), (0.5+0.5j)],
[(0.5+0.5j), (0.5-0.5j), (0.5-0.5j), (0.5+0.5j)]]

Dopo la conversione dei controlli di tipo '0':

A = [['*', '1', '1', 'T'],
['*', 'X', '*', '*'],
['T', '1', '*', '1'],
['*', 'X', '*', '*']]

V = [[(0.5+0.5j), (0.5-0.5j), (0.5-0.5j), (0.5+0.5j)],
[(0.5+0.5j), (0.5-0.5j), (0.5-0.5j), (0.5+0.5j)]]

Figura 4.3: Esempio di trasformazione di controlli da tipo '0' in controlli di tipo '1'.

Nella Figura 4.4 è riportato il diagramma di flusso del secondo step dell'algoritmo di Nielsen-Chuang. In questo step viene implementata la vera e propria scomposizione ricorsiva. Come si può notare dal diagramma di flusso, l'algoritmo scorre tutto lo schematico A linea per linea, alla ricerca di porte con un numero di controlli maggiore di uno. Qualora si trovi una porta con questa caratteristica, essa viene scomposta secondo l'algoritmo di Nielsen-Chuang, modificando lo schematico A e le relative linee della lista di lista di V. Nell'implementazione di questa funzione è stata lasciata la possibilità all'utente di scegliere se scomporre la porta Toffoli oppure no. Questa, infatti, è l'unica porta con due controlli che può essere descritta in OpenQASM 2.0; nonostante ciò in alcune situazioni può essere conveniente scomporre tale porta, per esempio nel caso in cui il simulatore su cui sarà eseguito l'OpenQASM 2.0 generato non supporti tale porta. La scelta di scomporre la porta Toffoli può essere fatta dall'utente tramite l'apposito parametro di ingresso alla funzione `toffoli`; questo flag deve essere posto a 1 se la porta Toffoli può essere utilizzata nel circuito finale descritto in OpenQASM 2.0, quindi senza essere scomposta, viceversa deve essere posto a 0 per scomporre la porta Toffoli.

Nella Figura 4.5 è riportato il diagramma di flusso della scomposizione di Nielsen-Chuang di una linea dello schematico A. Le nuove linee dello schematico vengono inserite in A_{new} e gli elementi delle relative matrici in V_{new}. Queste due liste di liste andranno a sostituire le vecchie linee dello schematico A e lista di liste V della porta appena scomposta.

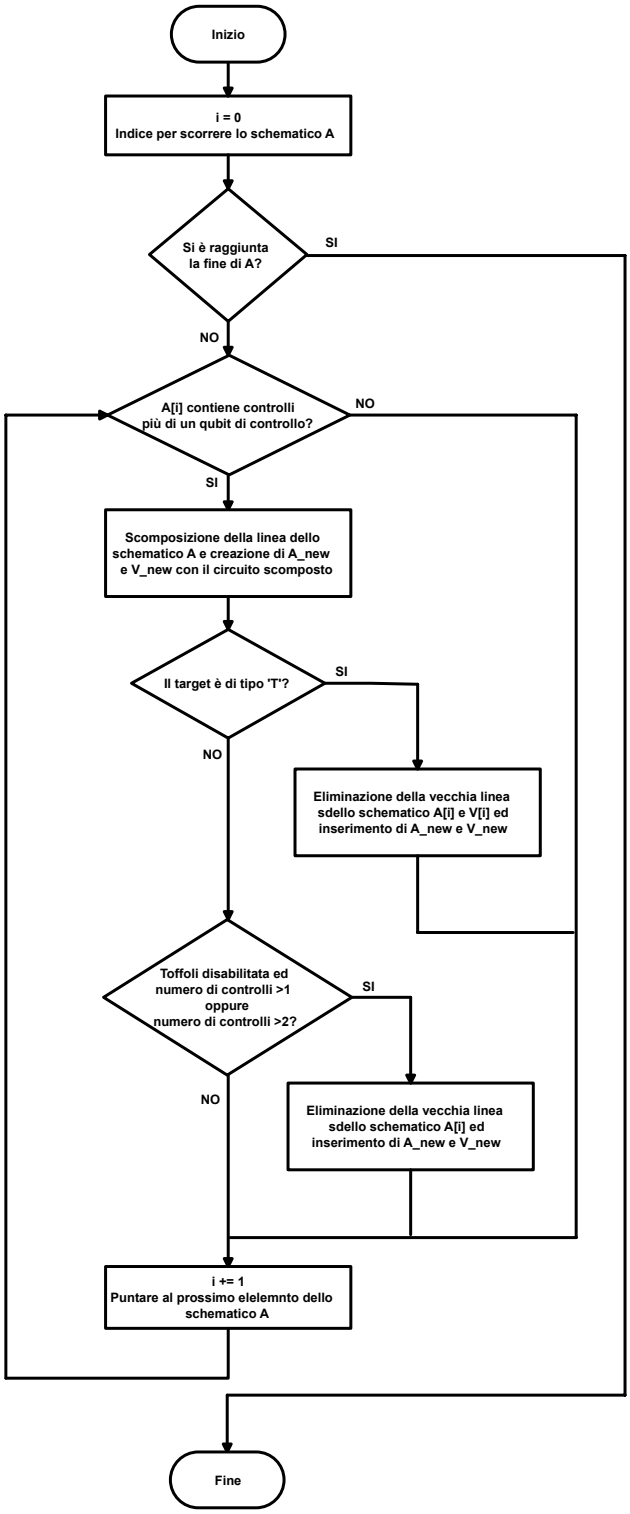


Figura 4.4: Diagramma di flusso della scomposizione ricorsiva di Nielsen-Chuang.

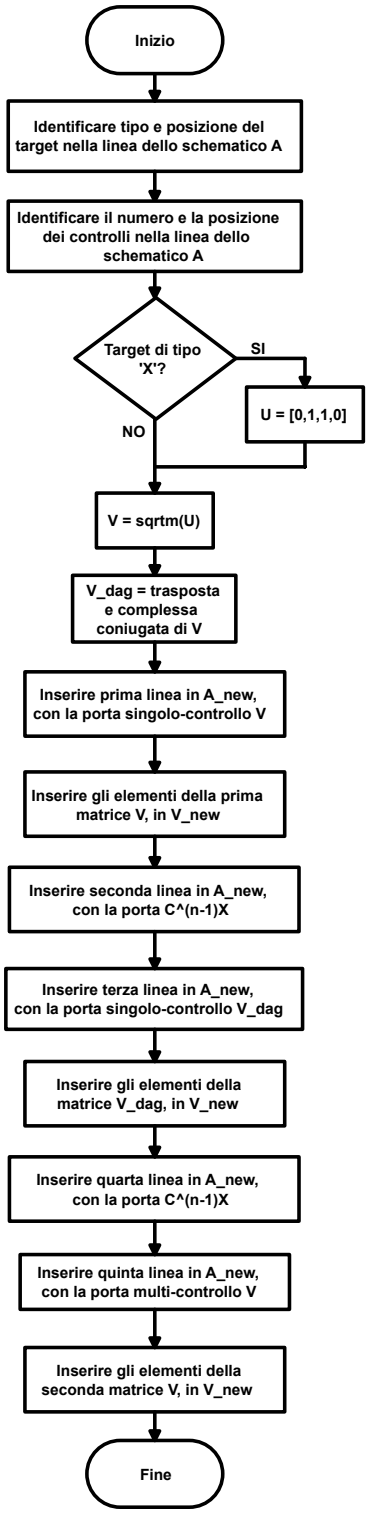


Figura 4.5: Diagramma di flusso della scomposizione di una linea dello schematico secondo l'algoritmo di Nielsen-Chuang.

4.2.2 Implementazione dell'algoritmo di generazione dell'OpenQASM 2.0 e recupero di fase

In questa sotto-sezione sono riportati i diagrammi di flusso usati per l'implementazione delle funzioni per la scrittura dei circuiti in OpenQASM 2.0 e per la procedura di recupero di fase.

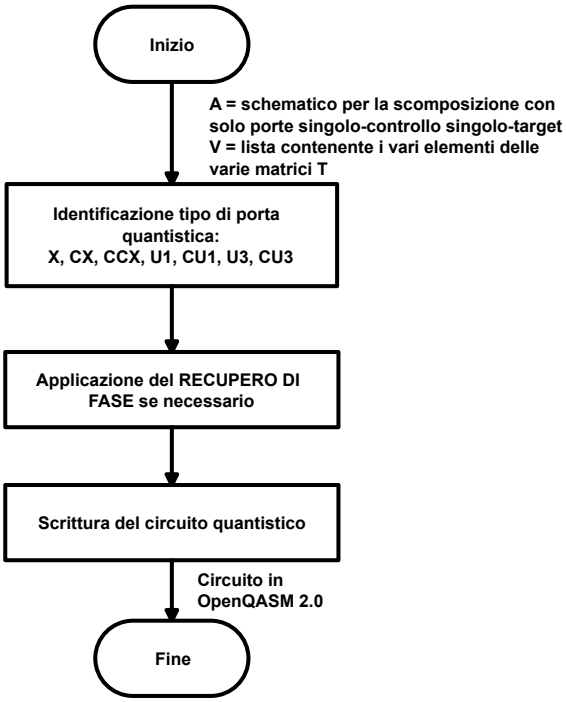


Figura 4.6: Diagramma di flusso della procedura di analisi di ogni porta dello schematico A per la scrittura dei circuiti quantistici in OpenQASM 2.0.

Nella Figura 4.6 è riportato il diagramma di flusso teorico della procedura di analisi di ogni porta per la scrittura dei circuiti quantistici in OpenQASM 2.0, partendo dallo schematico A e lista di liste V in uscita dall'algoritmo di Nielsen-Chuang. Per ogni linea dello schematico A viene identificato il tipo di porta quantistica da descrivere; l'algoritmo distingue tra sette porte: X, CX, CCX, $U1$, $CU1$, $U3$ e $CU3$. Se necessario, si applica il recupero di fase alla porta appena analizzata, infine viene scritto il circuito in OpenQASM 2.0.

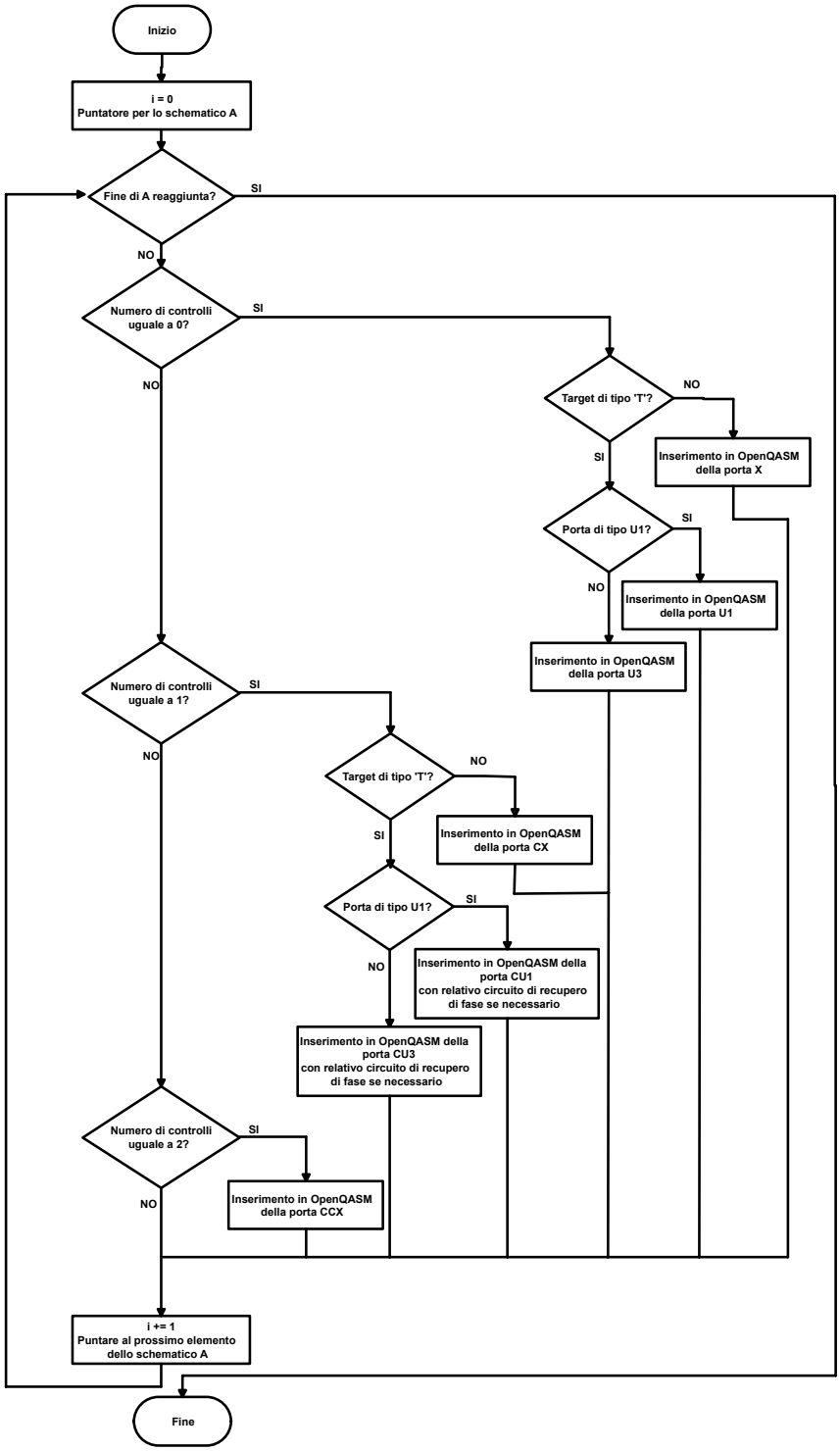


Figura 4.7: Diagramma di flusso per l'implementazione della funzione per la scrittura dei circuiti quantistici in OpenQASM 2.0.

Nella Figura 4.7 è riportato il diagramma di flusso usato per l'implementazione della funzione per la scrittura dei circuiti quantistici in OpenQASM 2.0 partendo dallo schematico A e lista di lista V in uscita dall'algoritmo di Nielsen-Chuang.
Per ogni linea dello schematico A viene identificato il tipo di porta quantistica, discriminando in primis il numero di controlli che essa presenta. Quindi si valuta se il target è di tipo `'T'` o `'X'` ed infine, in caso di target di tipo `'T'`, si stabilisce se la rotazione del target è una rotazione di tipo $U1$ o se è una rotazione di tipo $U3$.
In caso di rotazioni $CU1$ o $CU3$ potrebbe essere necessario l'applicazione del recupero di fase per descrivere correttamente la porta quantistica in maniera compatibile con OpenQASM 2.0.

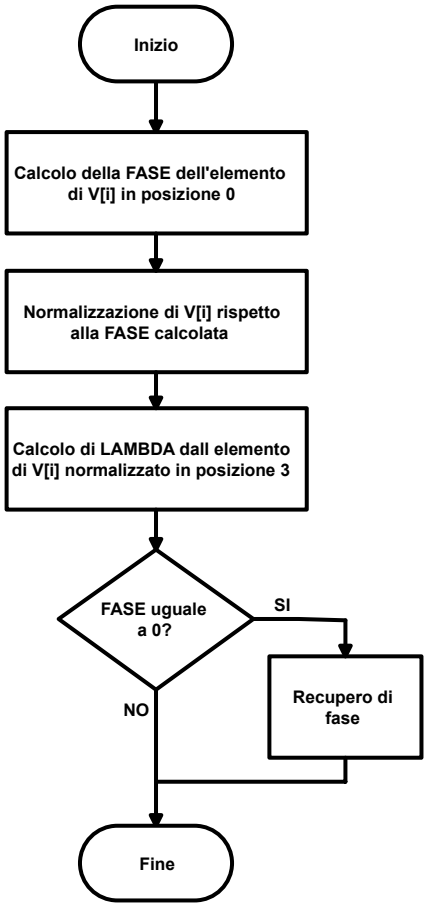


Figura 4.8: Diagramma di flusso per l'implementazione della funzione per la scrittura di una porta $CU1$ in OpenQASM 2.0.

Nella Figura 4.8 è riportato il diagramma di flusso per la scrittura di una porta di tipo $CU1$ in OpenQASM 2.0. E' utile osservare che nel caso di una porta non controllata di tipo $U1$ il diagramma di flusso è lo stesso ad esclusione del recupero di fase, che, in questo caso, non deve mai essere effettuato.

Nella Figura 4.9 è riportato il diagramma di flusso per la scrittura di una porta di tipo $CU3$ in OpenQASM 2.0. E' utile osservare che nel caso di una porta non controllata di tipo $U3$ il diagramma di flusso è lo stesso ad esclusione del recupero di fase, che, in questo caso, non deve mai essere effettuato. Nell'implementazione della funzione, è stata lasciata la possibilità all'utente di abilitare o meno i test sulla porta descritta. Il test consiste nel verificare che con i θ, ϕ e λ, calcolati rispettivamente tramite gli elementi V[i][0], V[i][2] e V[i][1], si ottenga l'elemento V[i][3] a meno di una tolleranza (anch'essa impostata dall'utente). La formula utilizzata nel test è la seguente:

$$V[i][3] = e^{j(\lambda+\phi)} \cos\left(\frac{\theta}{2}\right). \tag{4.14}$$

Nella Figura 4.10 è riportato il diagramma di flusso usato per l'implementazione della funzione per il recupero di fase. Come è possibile notare dalla figura, l'algoritmo del recupero di fase consiste nell'inserire le porte necessarie, come indicato nella sezione teorica. E' importante sottolineare che il termine λ delle porte $CU1$ addette al recupero di fase, nel diagramma di flusso è indicato come *FASE*, che fa riferimento alla fase del elemento V[i][0], calcolata nel diagramma di flusso in Figura 4.8 per le porte di tipo $CU1$ e in Figura 4.9 per le porte di tipo CU3.

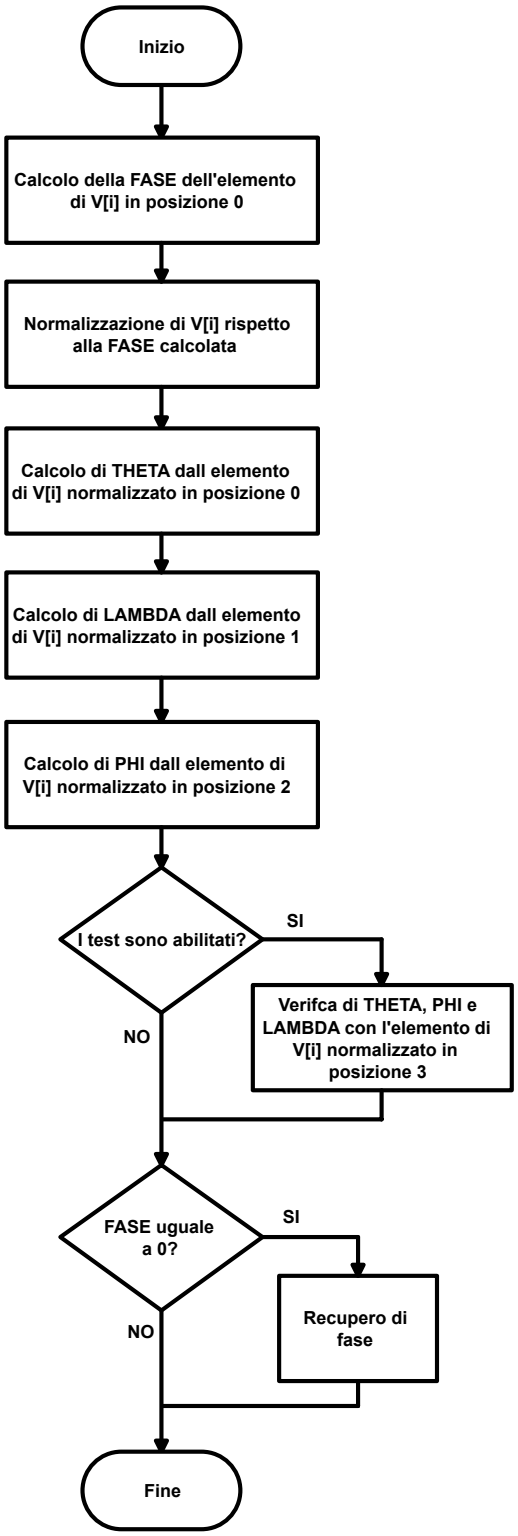


Figura 4.9: Diagramma di flusso per l'implementazione della funzione per la scrittura di una porta $CU3$ in OpenQASM 2.0.

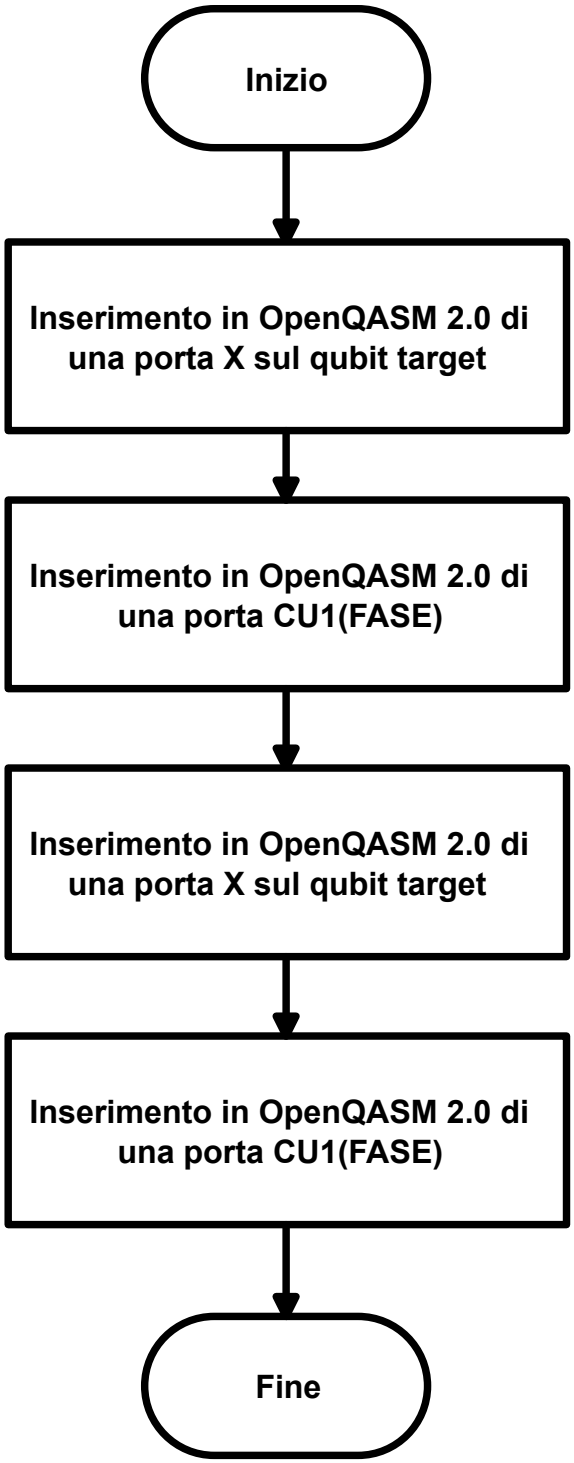


Figura 4.10: Diagramma di flusso per l'implementazione del recupero di fase.

4.3 Risultati

In questa sezione sono riportati i risultati di scomposizione secondo l'algoritmo ricorsivo di Nielsen-Chuang e la routine di scrittura dell'OpenQASM 2.0 con relativo recupero di fase. Le matrici utilizzate per il test sono le stesse adoperate per la scomposizione di Li-Pelejo nel capitolo 3. Si tratta di matrici casuali piene, quindi ogni elemento della matrice ha una parte reale e una parte immaginaria diversa da 0. Questo è il worst-case per quanto riguarda il tempo di scomposizione, l'errore numerico e il numero di porte quantistiche utilizzato nella scrittura finale del circuito in OpenQASM 2.0.
I risultati riportati nei grafici sono stati ottenuto mediando dieci scomposizioni a parità di valore del numero di qubit.

4.3.1 Tempo di scomposizione

In questa sotto-sezione sono riportati i grafici relativi al tempo di scomposizione secondo l'algoritmo ricorsivo di Nielsen-Chuang ed anche la routine di scrittura del OpenQASM 2.0.

Tempo di scomposizione Nielsen-Chuang Nelle Figure 4.11 e 4.12 sono riportati i grafici con i tempi di scomposizione secondo l'algoritmo ricorsivo di Nielsen-Chuang, rispettivamente in scala lineare e semi-logaritmica. Nei due grafici sono confrontati il tempo di scomposizione rispetto al numero di qubit (da due a otto), considerando la scomposizione della porta Toffoli `(toffoli:0)` rispetto a non al caso in cui non venga scomposta `(toffoli:1)`. Per entrambe le curve i test sono abilitati.
E' possibile notare che all'aumentare del numero di qubit aumenta la discrepanza tra il tempo necessario all'algoritmo di Nielsen-Chuang quando la porta Toffoli viene scomposta rispetto a quando essa non viene scomposta. Questo è dovuto al fatto che all'aumentare del numero di qubit aumenta il numero di porte Toffoli utilizzato nelle scomposizioni ricorsive delle porte di tipo multi-controllo singolo-target.
E' molto importante notare che il tempo impiegato per la scomposizione ricorsiva di Nielsen-Chuang è molto inferiore rispetto a quello impiegato per la scomposizione di Li-Pelejo, soprattutto al crescere del numero di qubit.

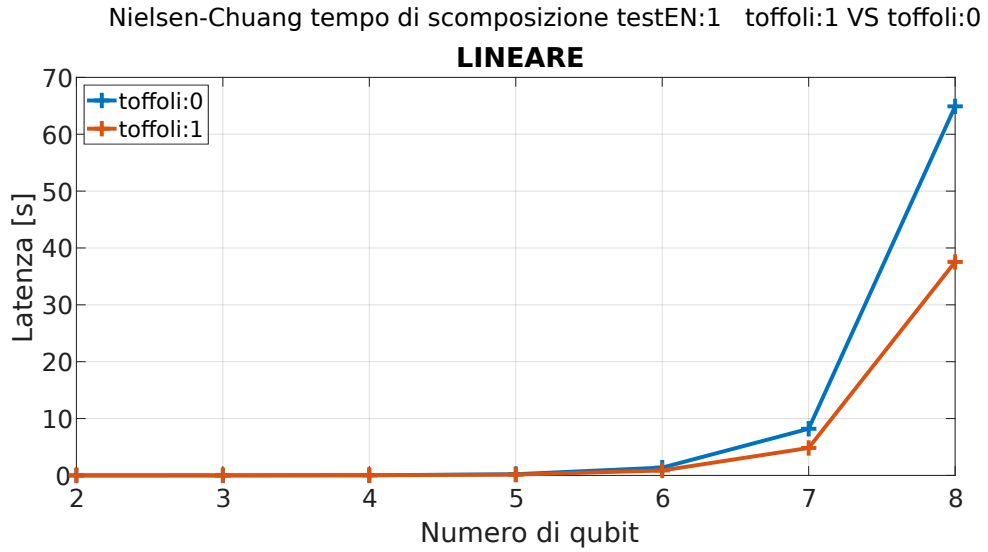


Figura 4.11: Grafico lineare di confronto del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno.

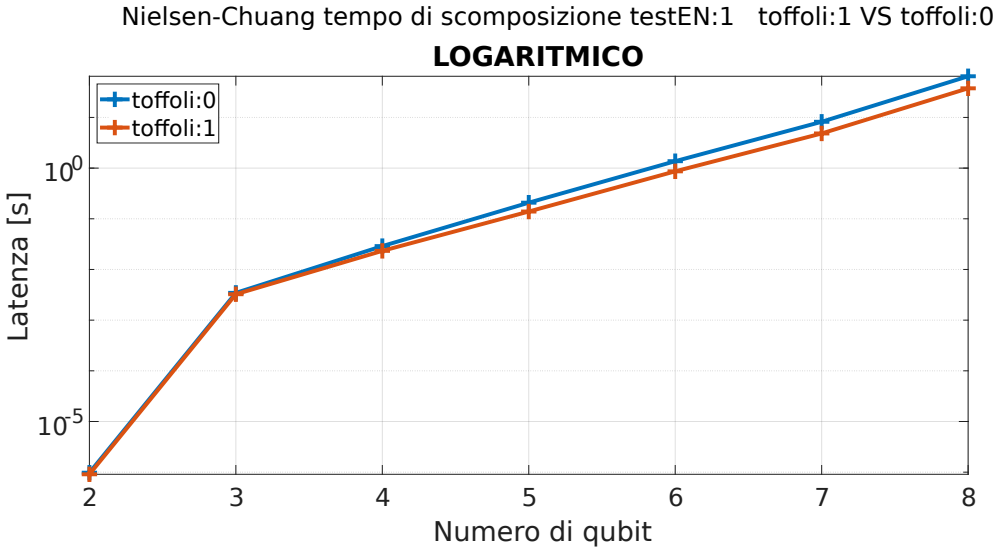


Figura 4.12: Grafico logaritmico di confronto del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno.

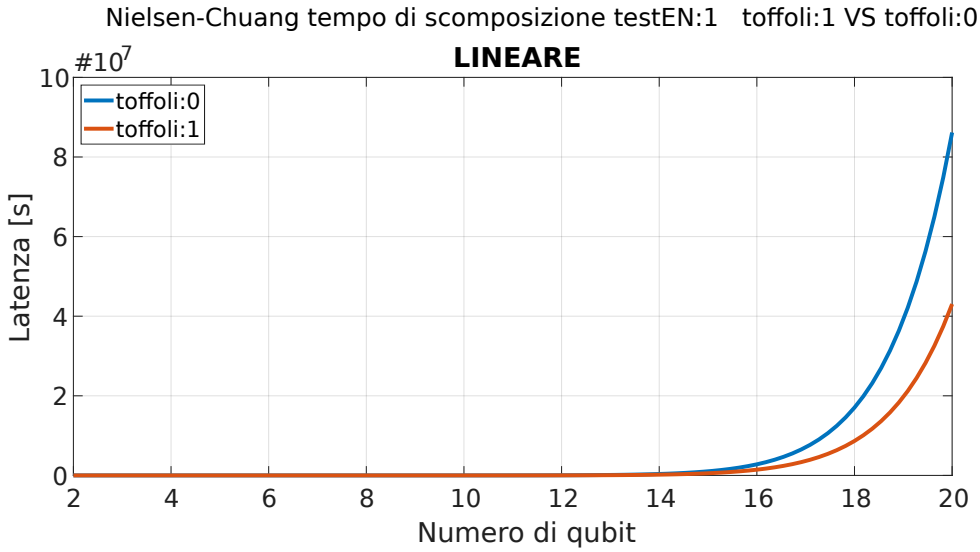


Figura 4.13: Grafico con l'interpolazione dei dati del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno, in scala lineare.

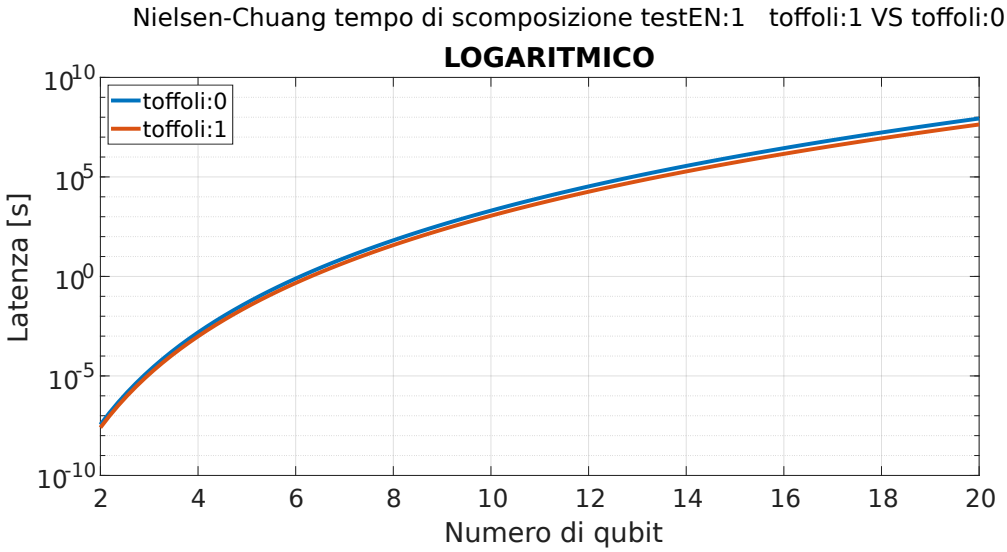


Figura 4.14: Grafico con l'interpolazione dei dati del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno, in scala semi-logaritmica.

Nelle Figure 4.13 e 4.14 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) con l'interpolazione del tempo di scomposizione dell'algoritmo di

Nielsen-Chuang esteso fino a venti qubit. Questi grafici sono da interpretare come una previsione del tempo necessario per la scomposizione. Anche in questi grafici si può notare l'aumento della differenza del tempo necessario per la scomposizione tra le due curve.

Nelle Figure 4.15 e 4.16 sono riportati i grafici di confronto per il tempo di scomposizione impiegato dall'algoritmo ricorsivo di Nielsen-Chuang all'aumentare del numero di qubit (da due a otto), comparando il caso in cui i test sono abilitati rispetto a quando non lo siano. Per entrambe le curve si è scelto di rappresentare la condizione peggiore per il tempo di scomposizione, ovvero abilitando la scomposizione della porta Toffoli.
E' possibile osservare che all'aumentare del numero di qubit aumenta la differenza tra le due curve. In questo caso, rispetto a quanto accade con la scomposizione di Li-Pelejo, la differenza di tempo, tra la curva relativa ai test abilitati e la curva relativa ai test non abilitati, non è cosi marcata; questo è dovuto al fatto che nel caso dei test nell'algoritmo di Nielsen-Chuang, essi vengono eseguiti una volta al fondo della scomposizione, verificando che tutte le matrici contenute nella lista di liste V siano unitarie. I test in questo caso sono test su matrici 2×2 indipendentemente dal numero di qubit, per cui l'aumento di tempo è dovuto solamente all'aumento del numero di matrici salvate all'interno di V, differentemente dall'algoritmo di Li-Pelejo, dove il test dipende sia dall'aumento delle matrici, sia dall'aumento delle dimensioni delle matrici.

Nelle Figure 4.17 e 4.18 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) con l'interpolazione dei dati del tempo si scomposizione dell'algoritmo ricorsivo di Nielsen-Chuang, confrontando i due casi con i test abilitati o disabilitati. Il range del numero di qubit è esteso fino a venti qubit. Anche questi grafici mostrano un aumento della differenza di latenza con e senza i test abilitati.

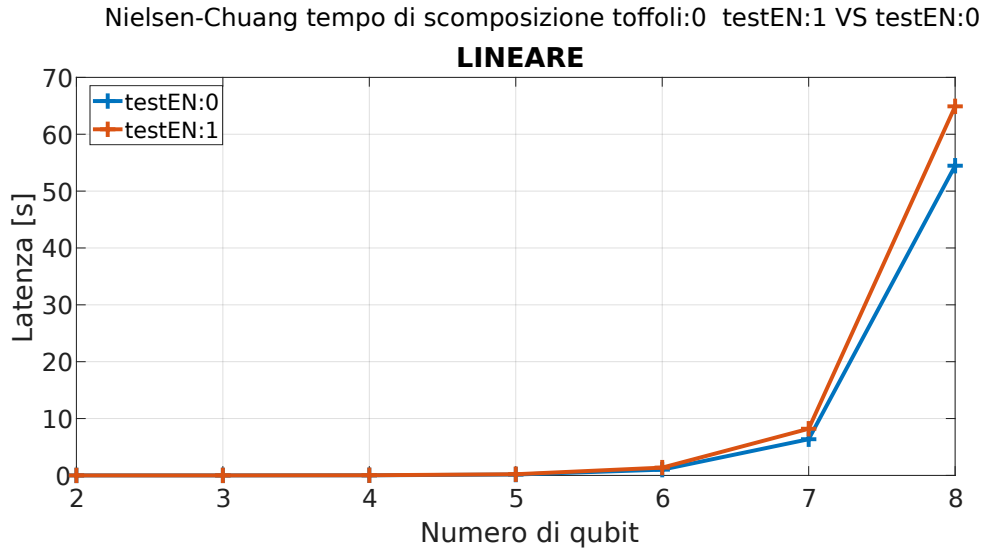


Figura 4.15: Grafico lineare di confronto del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con scomposizione della porta Toffoli quando i test sono abilitati o meno.

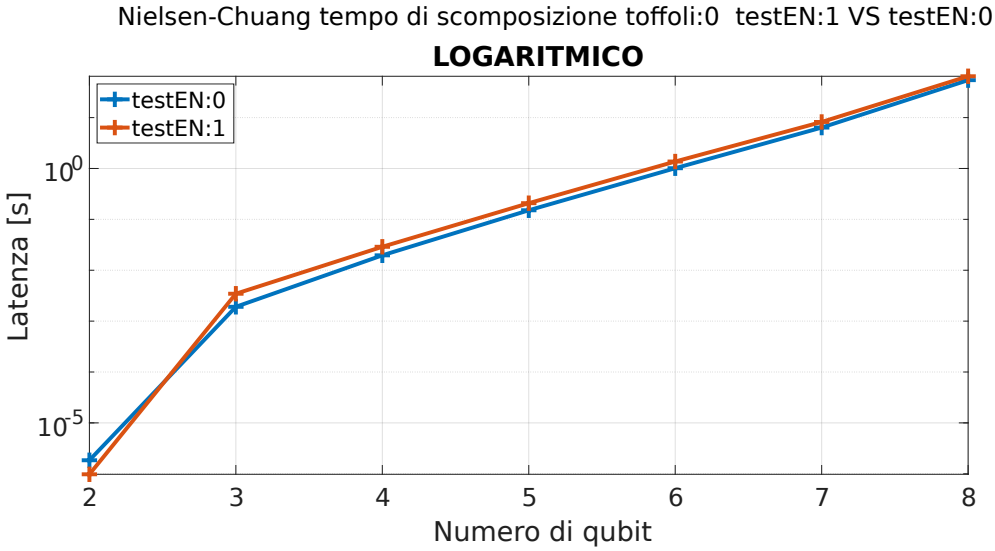


Figura 4.16: Grafico logaritmico di confronto del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con scomposizione della porta Toffoli quando i test sono abilitati o meno.

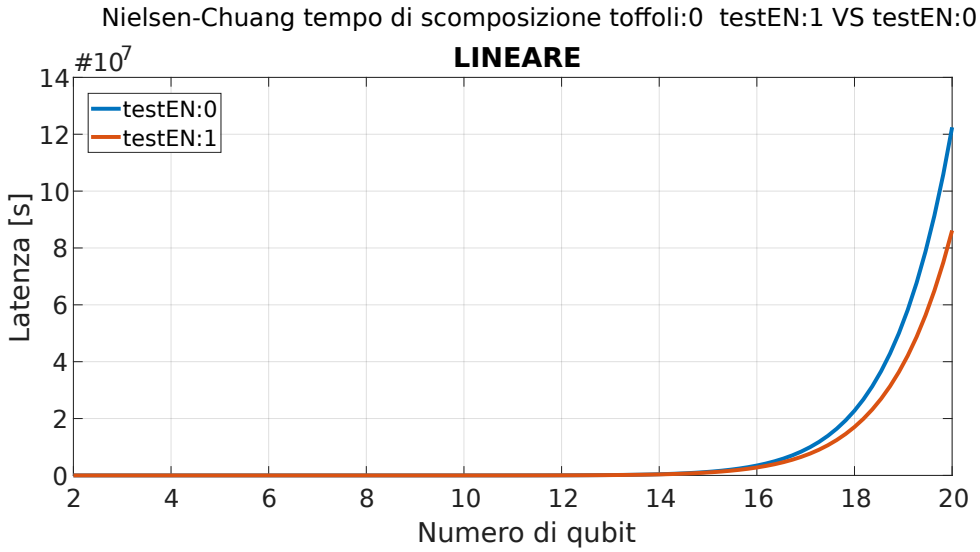


Figura 4.17: Grafico con l'interpolazione dei dati del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con scomposizione della porta Toffoli quando i test sono abilitati o meno, in scala lineare.

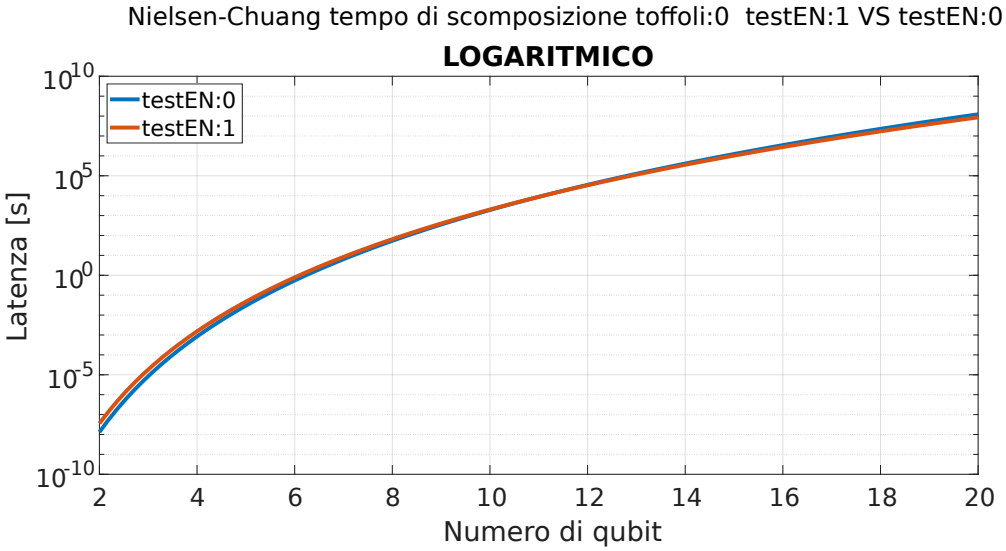


Figura 4.18: Grafico con l'interpolazione dei dati del tempo di scomposizione dell'algoritmo di Nielsen-Chuang con scomposizione della porta Toffoli quando i test sono abilitati o meno, in scala semi-logaritmica.

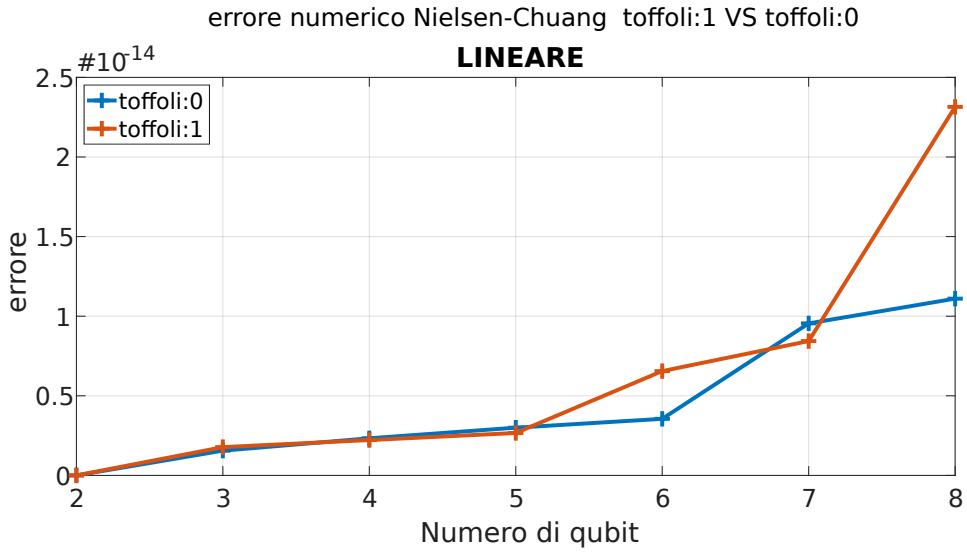


Figura 4.19: Grafico con l'errore numerico generato durante la scomposizione di Nielsen-Chuang in scala lineare.

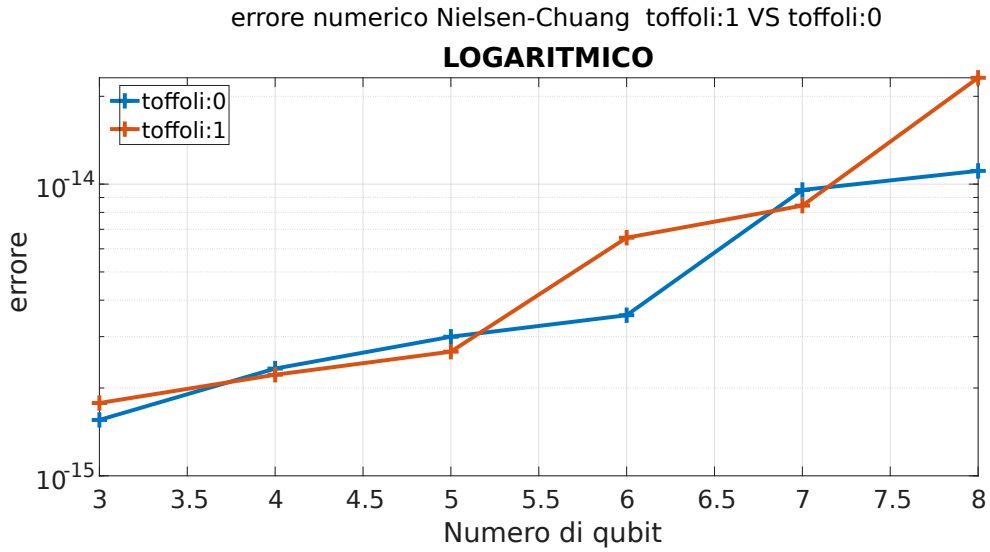


Figura 4.20: Grafico con l'errore numerico generato durante la scomposizione di Nielsen-Chuang in scala semi-logaritmica.

Errore numerico generato durante la scomposizione di Nielsen-Chuang Nelle Figure 4.19 e 4.20 sono riportati i grafici (rispettivamente in scala lineare e logaritmica) dell'errore numerico generato durante la scomposizione di Nielsen-Chuang. Per errore numerico si intende la distanza tra la moltiplicazione delle matrici 2×2 contenute nella lista di liste V e la matrice identità $\mathbf{I}_2$. Si può notare che all'aumentare del numero di qubit aumenta l'errore generato. Questo è dovuto al fatto che aumenta il numero di volte in cui deve essere effettuata ricorsivamente l'operazione di radice quadrata matriciale, aumenta linearmente all'aumentare del numero di qubit.

Tempo di scomposizione scrittura OpenQASM e recupero di fase Nelle Figure 4.21 e 4.22 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del tempo necessario per la scrittura del circuito quantistico in OpenQASM 2.0, con relativa procedura di recupero di fase, partendo dall'uscita della funzione di scomposizione di Nielsen-Chuang. Nei grafici sono confrontati i tempi nei casi in cui la porta Toffoli venga scomposta `(toffoli:0)` rispetto a quando essa non viene scomposta `(toffoli:1)`. Per entrambe le curve i test sono abilitati.
E' possibile notare come all'aumentare del numero di qubit aumenti la differenza di tempo tra le due curve. Questo perché nel caso di scomposizione della porta Toffoli, aumenta il numero di porte da tradurre in OpenQASM 2.0, aumentando anche i casi in cui è necessario applicare il recupero di fase. Infatti, come spiegato nella sezione della teoria, la scomposizione della porta Toffoli genera la porta controllate con rotazione del target $X^{\frac{1}{2}}$ (ed anche $X^{-\frac{1}{2}}$), che corrisponde alla matrice:

$$X^{\frac{1}{2}} = \begin{bmatrix} 0.5 + 0.5j & 0.5 - 0.5j \\ 0.5 - 0.5j & 0.5 + 0.5j \end{bmatrix} . \tag{4.15}$$

La fase dell'elemento (0,0) è 0.785 che è diversa da 0, per cui queste porte necessiteranno della procedura del recupero di fase.

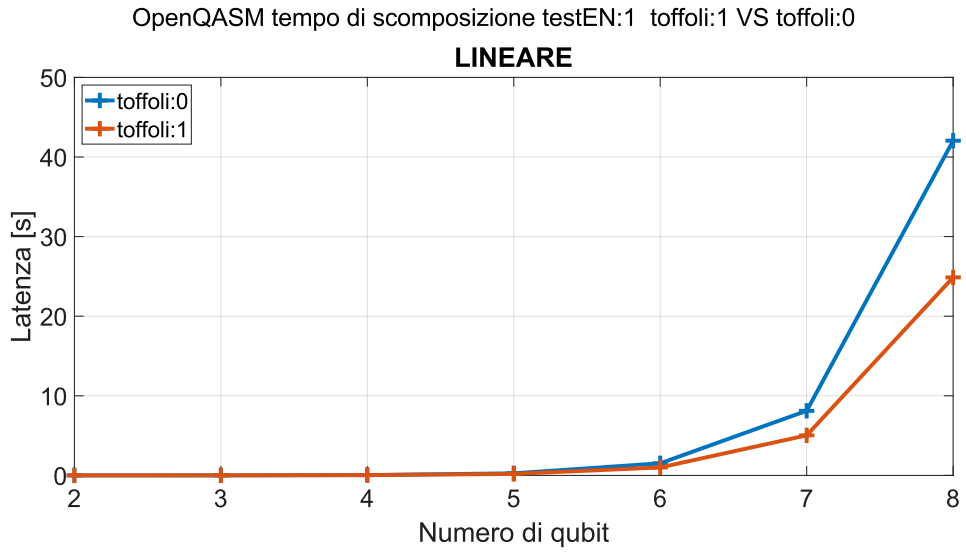


Figura 4.21: Grafico lineare di confronto del tempo di scrittura dell'OpenQASM e recupero di fase con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno.

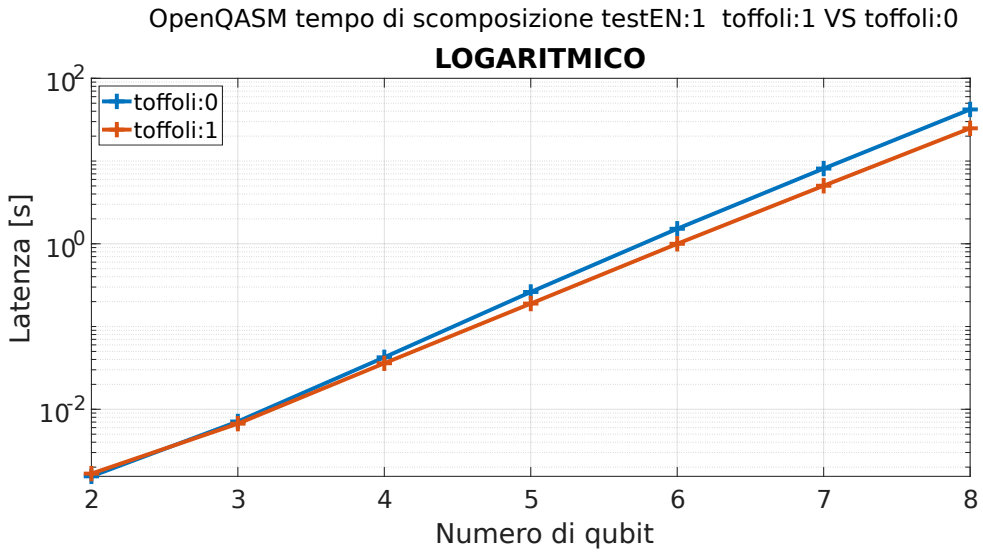


Figura 4.22: Grafico logaritmico di confronto del tempo di scrittura dell'OpenQASM e recupero di fase con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno.

Nelle Figure 4.23 e 4.24 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) riguardanti l'interpolazione dei dati del tempo necessario per la generazione del circuito quantistico in OpenQASM 2.0 con relativo recupero di fase. Il range del numero di qubit è esteso fino a venti qubit.
Anche in questi grafici è possibile osservare l'aumento della differenza dei tempi in caso di scomposizione della porta Toffoli rispetto al caso senza la scomposizione della porta Toffoli.

Nelle Figure 4.25 e 4.26 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del tempo necessario per la scrittura del circuito quantistico in OpenQASM 2.0, con relativa procedura di recupero di fase, partendo dalle uscita della funzione di scomposizione di Nielsen-Chuang. Nei grafici sono confrontati i tempi nei casi in cui i test siano abilitati `(testEN:1)` o disabilitati `(testEN:0)`. Per entrambe le curve la porta Toffoli viene scomposta.
E' possibile notare l'aumento, all'aumentare del numero di qubit, della differenza di tempo tra i casi con e senza test. In questo caso la differenza dei tempi è inferiore rispetto a quello della scomposizione di Nielsen-Chuang, in quanto i test vengono effettuati solamente sulle porte con rotazione del target di tipo $U3$, come spiegato nella sezione di implementazione dell'algoritmo del recupero di fase in questo capitolo, per cui l'aumento della differenza è legata all'aumento del numero di porte con rotazione del target di tipo $U3$.

Nelle Figure 4.27 e 4.28 sono riportati i grafici con l'interpolazione dei dati (rispettivamente in scala lineare e semi-logaritmica) del tempo necessario per la scrittura del circuito quantistico in OpenQASM 2.0, con relativa procedura di recupero di fase, partendo dalle uscita della funzione di scomposizione di Nielsen-Chuang; confrontando il caso di test abilitati o disabilitati. Il range del numero di qubit è esteso fino a venti qubit.
Anche questi grafici mostrano un aumento della differenza di latenza tra il caso di test abilitati rispetto a quello con test disabilitati.

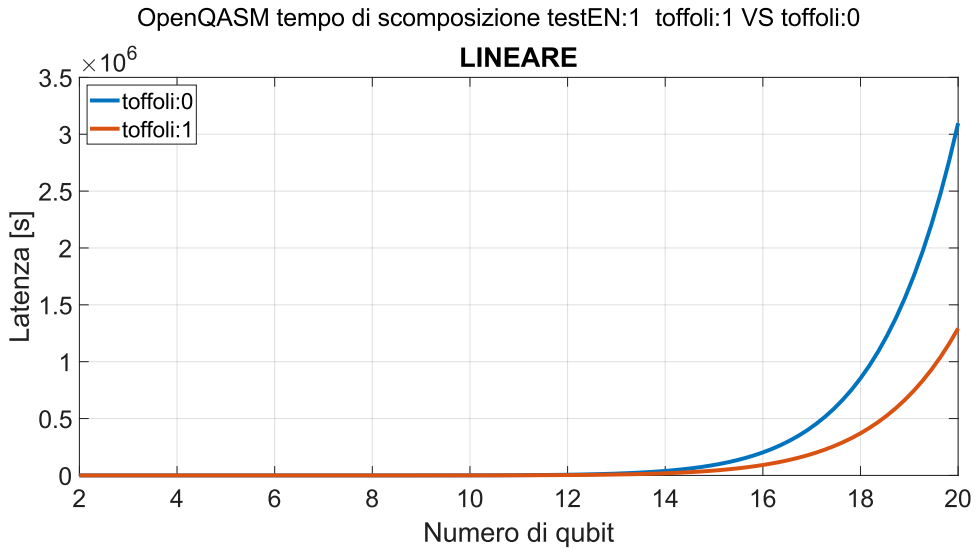


Figura 4.23: Grafico con l'interpolazione dei dati del tempo di scrittura dell'OpenQASM e recupero di fase con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno, in scala lineare.

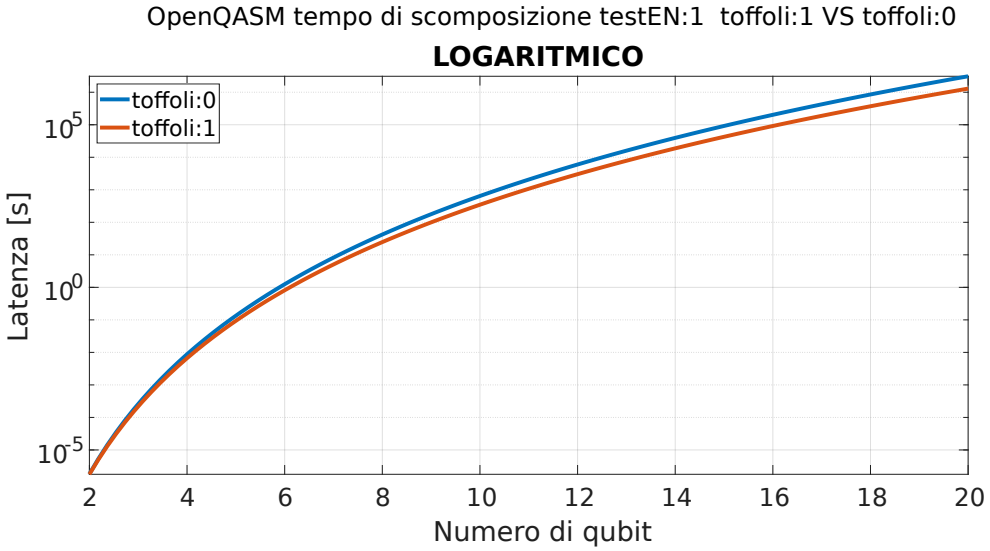


Figura 4.24: Grafico con l'interpolazione dei dati del tempo di scrittura dell'OpenQASM e recupero di fase con test abilitati quando la scomposizione della porta Toffoli e abilitata o meno, in scala semi-logaritmica.

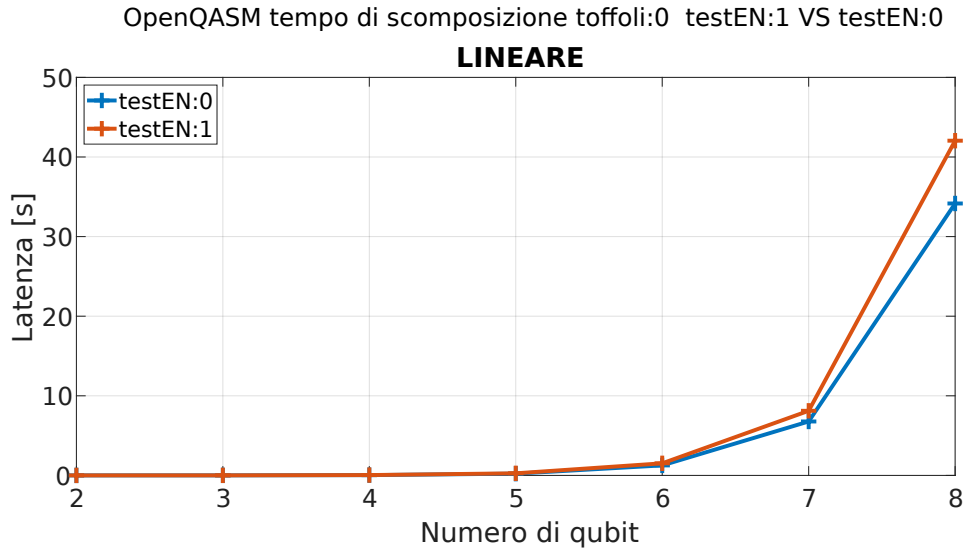


Figura 4.25: Grafico in scala lineare di confronto del tempo di generazione dell'OpenQASM 2.0 e relativo recupero di fase con scomposizione della porta Toffoli quando i test sono abilitati o meno.

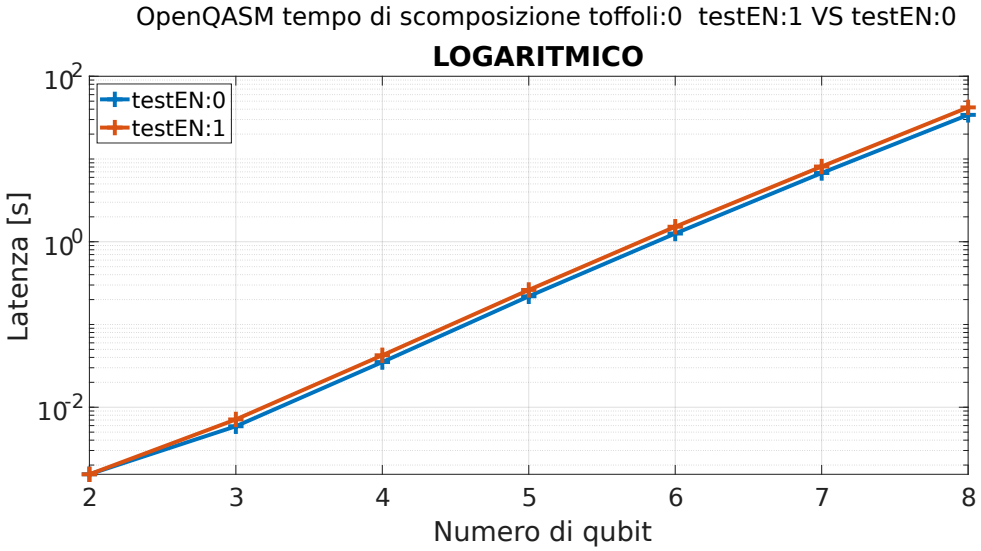


Figura 4.26: Grafico in scala semi-logaritmica di confronto del tempo di generazione dell'OpenQASM 2.0 e relativo recupero di fase con scomposizione della porta Toffoli quando i test sono abilitati o meno.

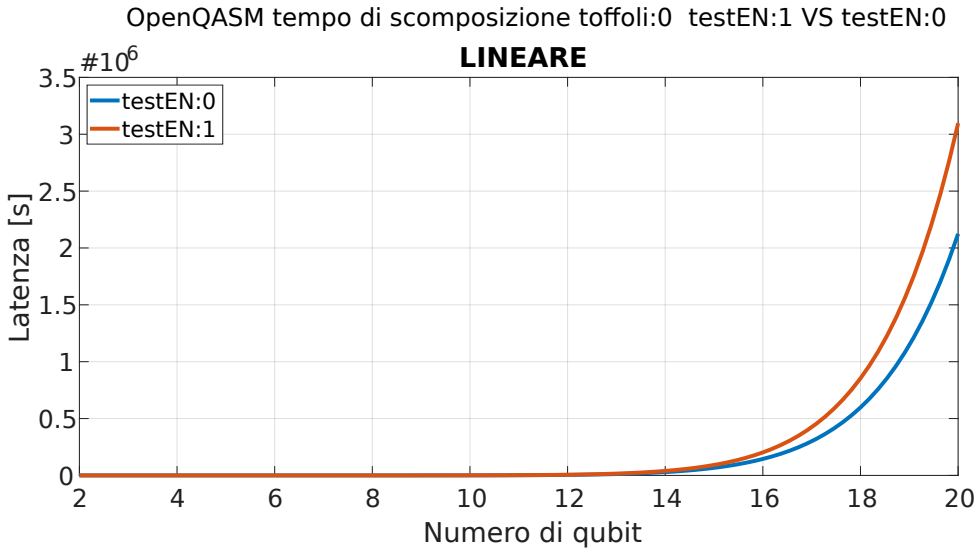


Figura 4.27: Grafico con l'interpolazione dei dati del tempo di generazione dell'OpenQASM 2.0 e relativo recupero di fase con scomposizione della porta Toffoli quando i test sono abilitati o meno, in scala lineare.

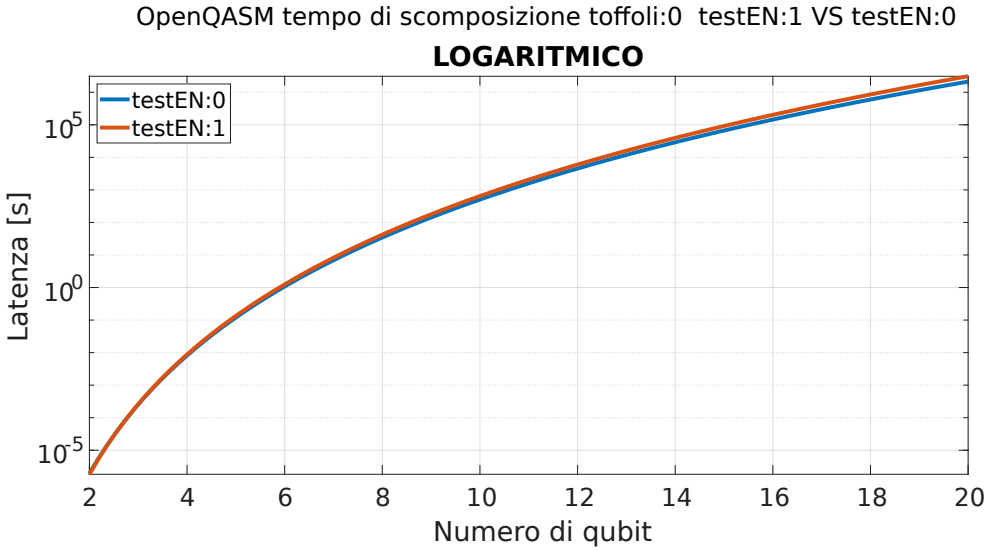


Figura 4.28: Grafico con l'interpolazione dei dati del tempo di generazione dell'OpenQASM 2.0 e relativo recupero di fase con scomposizione della porta Toffoli quando i test sono abilitati o meno, in scala semi-logaritmica.

4.3.2 Risultati complessivi della scomposizione

In questa sotto-sezione sono riportati i grafici con i risultati delle scomposizioni complessive: da matrici unitarie a descrizione del circuito quantistico, attraverso l'applicazione in sequenza degli algoritmi di Li-Pelejo, Nielsen-Chuang e scrittura del circuito in OpenQASM 2.0 con relativa procedura di recupero di fase.
Nei seguenti paragrafi vengono confrontati i risultati in termine di tempo di scomposizione, errore numerico e numero di porte quantistiche utilizzate per la descrizione del circuito in OpenQASM 2.0.

Tempo di scomposizione Nelle Figure 4.29 e 4.30 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) con il tempo totale di scomposizione per le matrici unitarie, attraverso l'esecuzione dell'algoritmo di scomposizione di Li-Pelejo, Nielsen-Chuang e scrittura in OpenQASM 2.0 con relativa procedura di recupero di fase. Nei grafici vengono confrontati i quattro casi possibili di scomposizione: test abilitati e porta Toffoli scomposta `(testEN:1, toffoli:0)`, test abilitati e porta Toffoli non scomposta `(testEN:1, toffoli:1)`, test disabilitati e porta Toffoli scomposta `(testEN:0, toffoli:0)` e test disabilitati e porta Toffoli non scomposta `(testEN:0, toffoli:1)`.
Come si può osservare dai grafici, il tempo di scomposizione dell'algoritmo di Li-Pelejo domina rispetto ai tempi degli altri due algoritmi utilizzati dopo di esso.
Inoltre, è possibile osservare che l'abilitazione o disabilitazione dei test influisce in maniera molto più significativa, rispetto all'abilitazione o disabilitazione della scomposizione della porta Toffoli. Questo fenomeno è dovuto al peso maggiore del tempo di scomposizione di Li-Pelejo, infatti l'abilitazione o la disabilitazione della porta Toffoli non incide sulle prestazioni dell'algoritmo di Li-Pelejo, mentre l'abilitazione o la disabilitazione dei test ne modifica pesantemente le prestazioni.
Il caso peggiore è quello in cui i test sono abilitati ed è abilitata anche la scomposizione della porta Toffoli. Invece il caso migliore è quello in cui sono disabilitati i test ed è disabilitata anche la scomposizione della porta Toffoli.

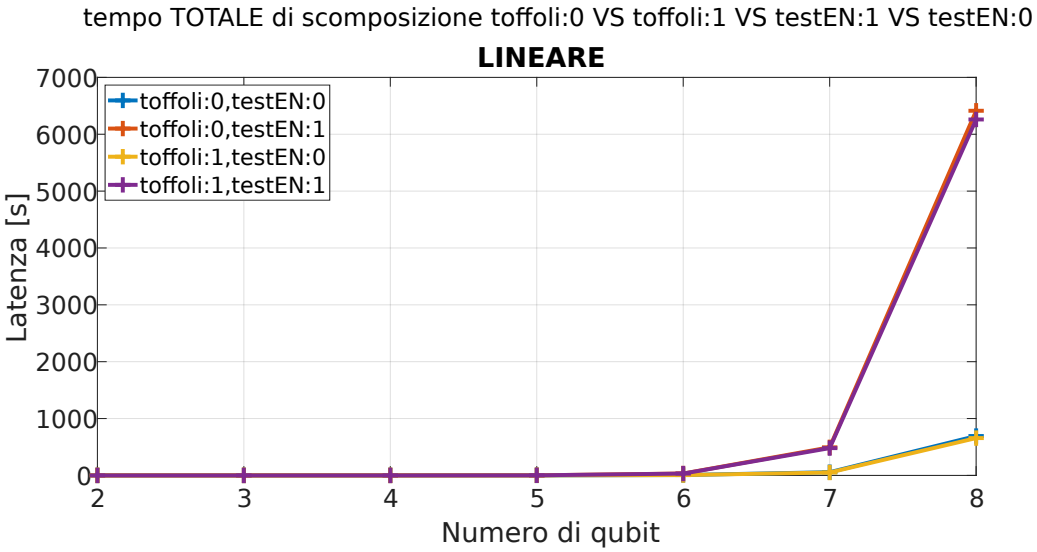


Figura 4.29: Grafico con il tempo totale di scomposizione di una matrice unitarie in scala lineare.

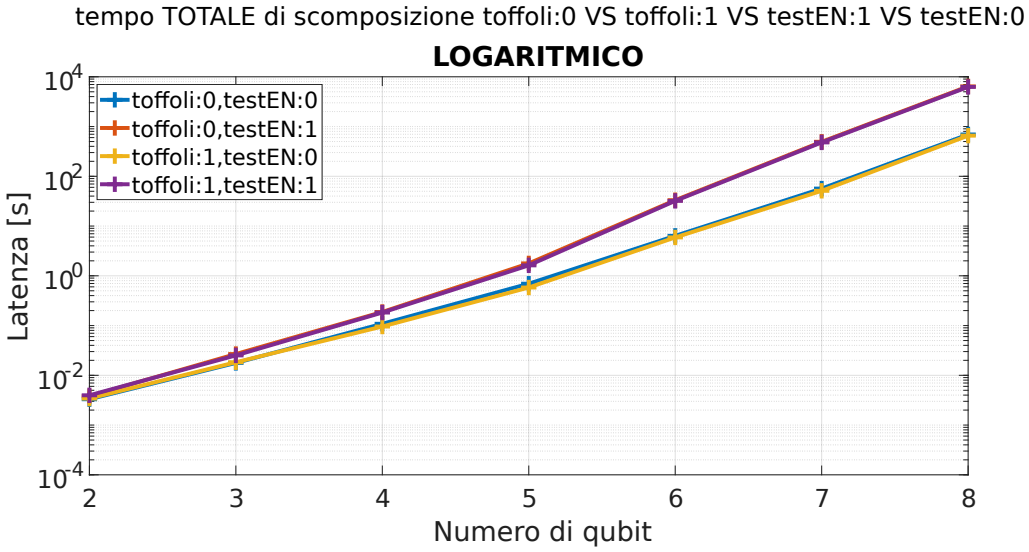


Figura 4.30: Grafico con il tempo totale di scomposizione di una matrice unitarie in scala semi-logaritmica.

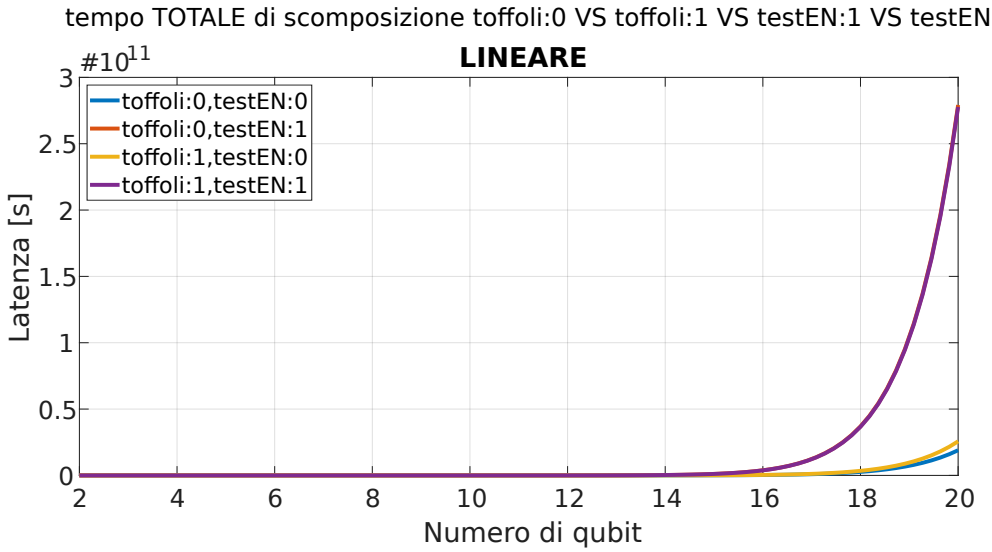


Figura 4.31: Grafico con l'interpolazione del tempo totale di scomposizione di una matrice unitarie, in scala lineare.

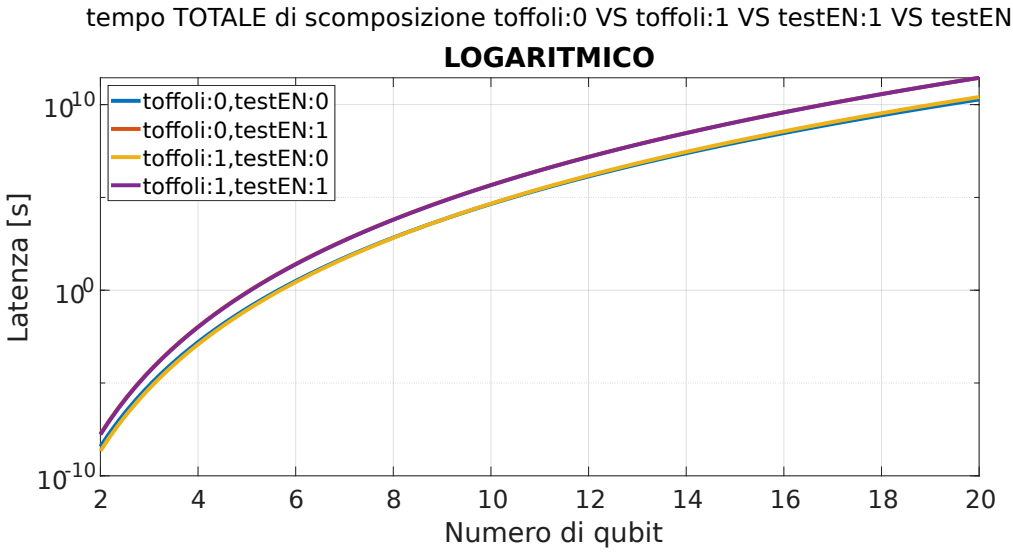


Figura 4.32: Grafico con l'interpolazione del tempo totale di scomposizione di una matrice unitarie, in scala semi-logaritmica.

Nelle Figure 4.31 e 4.32 riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) con l'interpolazione del tempo totale di scomposizione per le matrici unitarie, attraverso l'esecuzione dell'algoritmo di scomposizione di Li-Pelejo, Nielsen-Chuang e scrittura in OpenQASM 2.0 con relativa procedura di recupero di fase. Il range del numero di qubit è aumentato fino a venti qubit.
Anche in questo caso il caso peggiore è quello in cui i test sono abilitati ed è abilitata anche la scomposizione della porta Toffoli. Invece il caso migliore è quello in cui sono disabilitati i test ed è disabilitata anche la porta scomposizione della porta Toffoli.
Questi grafici sono da intendere come una previsione del tempo di scomposizione totale di una matrice unitaria al variare del numero di qubit.

Errore numerico Nelle Figure 4.33 e 4.34 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) con il confronto dell'errore numerico generato dagli algoritmi di scomposizione di Li-Pelejo e Nielsen-Chuang.
Sono rappresentati gli errori nel caso in cui la porta Toffoli sia scomposta, caso peggiore per l'algoritmo di Nielsen-Chuang.
Come è possibile notare dai grafici l'errore di Li-Pelejo domina rispetto a quello di Nielsen-Chuang, soprattutto all'aumentare del numero di qubit. Questo fenomeno è principalmente dovuto al modo con cui vengono calcolati gli errori numerici; infatti, nel caso di Li-Pelejo il calcolo viene fatto su matrici di dimensioni $2^n \times 2^n$ (dove n rappresenta il numero di qubit), quindi la moltiplicazione riga per colonna delle matrici interessa un numero elevato di valori. Invece, nel caso di Nielsen-Chuang l'errore viene calcolato su matrici 2×2, indipendentemente dal numero di qubit, per cui le moltiplicazioni riga colonna operano su un numero molto ridotto di valori.
E' importante evidenziare che i due errori non sono scorrelati; infatti, le matrici di partenza per l'algoritmo di Nielsen-Chuang sono generate dall'algoritmo di Li-Pelejo, per cui l'andamento crescente dell'errore dell'algoritmo di Nielsen-Chuang è in parte dovuto all'aumento del errore "ereditato" da Li-Pelejo.

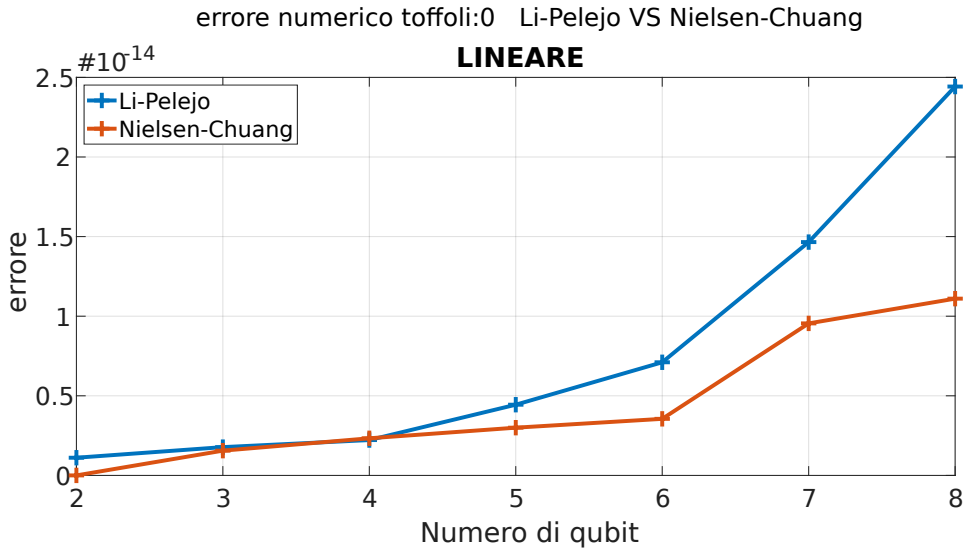


Figura 4.33: Grafico con il confronto dell'errore numerico generato dagli algoritmi di scomposizione di Li-Pelejo e Nielsen-Chuang in scala lineare.

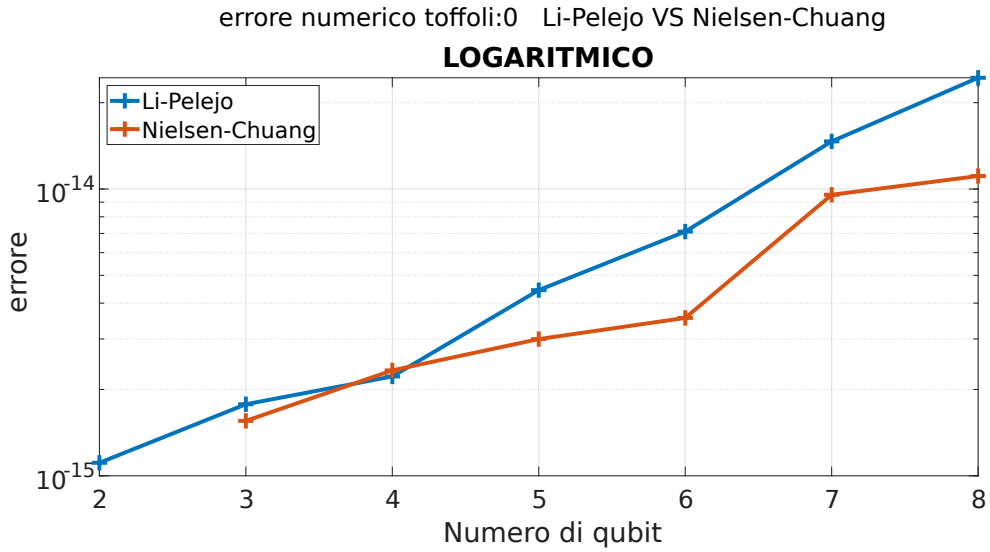


Figura 4.34: Grafico con il confronto dell'errore numerico generato dagli algoritmi di scomposizione di Li-Pelejo e Nielsen-Chuang in scala semi-logaritmica.

Numero di porte quantistiche utilizzato per la descrizione del circuito in OpenQASM 2.0 Nelle Figure 4.35 e 4.36 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) con il confronto del numero di porte quantistiche per la descrizione del circuito in OpenQASM 2.0, con e senza l'abilitazione della scomposizione della porta Toffoli (`toffoli:0` e `toffoli:1`).
E' possibile osservare come nel caso di abilitazione della scomposizione della porta Toffoli aumenti il numero di porte complessive utilizzate. Quest'aumento è sempre maggiore con l'aumentare del numero di qubit.
Il numero di porte utilizzato dipende principalmente dall'algoritmo di Nielsen-Chuang e dalla procedura del recupero di fase; Infatti ogni porta scomposta con Nielsen-Chuang genera cinque porte ad ogni iterazione della scomposizione. Inoltre, il metodo del recupero di fase inserisce quattro porte in più per ogni porta quantistica che richiede tale procedura per essere descritta correttamente in OpenQASM 2.0.
E' importante sottolineare che il meccanismo di scomposizione di Nielsen-Chuang è "abilitato" dal numero di controlli per ogni singola porta generati dall'algoritmo di Li-Pelejo; per cui per ridurre il numero di porte si potrebbe indagare un altro algoritmo da utilizzare al posto dell'algoritmo di Li-Pelejo.

Nelle Figure 4.31 e 4.32 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) con l'interpolazione dei dati del confronto di numero di porte quantistiche utilizzato per la descrizione del circuito in OpenQASM 2.0 in caso di abilitazione della scomposizione della porta Toffoli (`toffoli:0`) rispetto al caso in cui essa non sia scomposta (`toffoli:1`). Il range del numero di qubit è aumentato fino a venti. Le curve in questi grafici sono da intendere come una previsione del numero di porte quantistiche utilizzato per la descrizione del circuito in OpenQASM 2.0.

Tabelle riassuntive Nella Tabella 4.1 vengono riassunti i dati riportati nei vari grafici relativi ai tempi di ogni algoritmo utilizzato per la scomposizione delle matrici. Nella Tabella 4.2 vengono riassunti i dati relativi ai tempi totali per scomposizione delle matrici. Nella Tabella 4.3 vengono riassunti i dati relativi al errore numerico generato dagli algoritmi di Li-Pelejo e Nielsen-Chuang. Nella Tabella 4.4 vengono riassunti i dati relativi al numero di porte adoperate per la creazione dei

circuiti in OpenQASM 2.0.

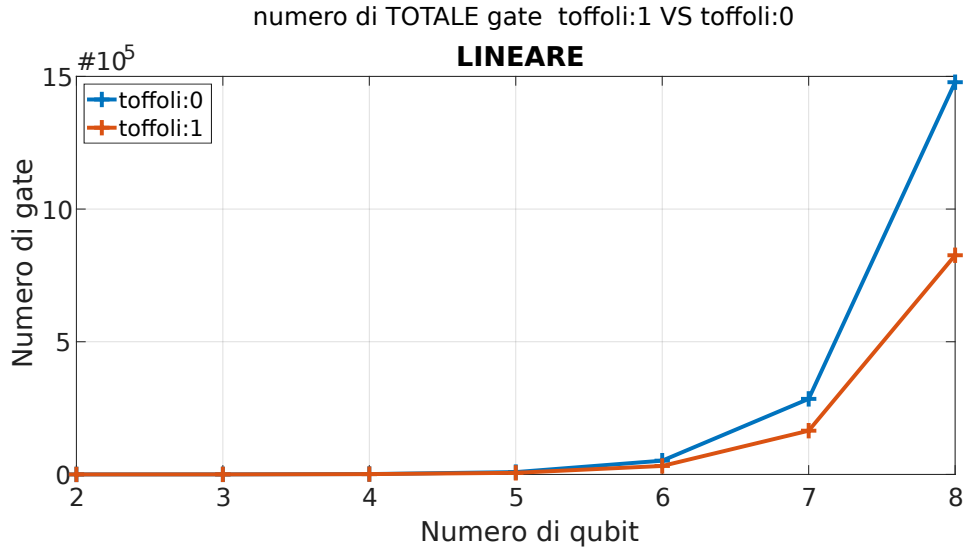


Figura 4.35: Grafico con il confronto del numero di porte quantistiche utilizzato per la descrizione del circuito in OpenQASM 2.0 in caso di abilitazione della scomposizione della porta Toffoli `(toffoli:0)` o meno `(toffoli:1)` in scala lineare.

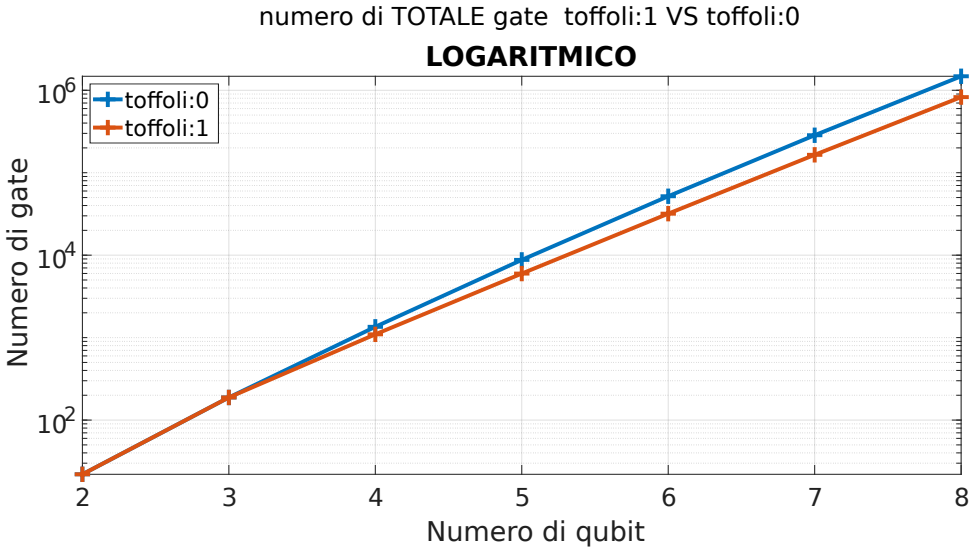


Figura 4.36: Grafico con il confronto del numero di porte quantistiche utilizzato per la descrizione del circuito in OpenQASM 2.0 in caso di abilitazione della scomposizione della porta Toffoli `(toffoli:0)` o meno `(toffoli:1)` in scala semi-logaritmica.

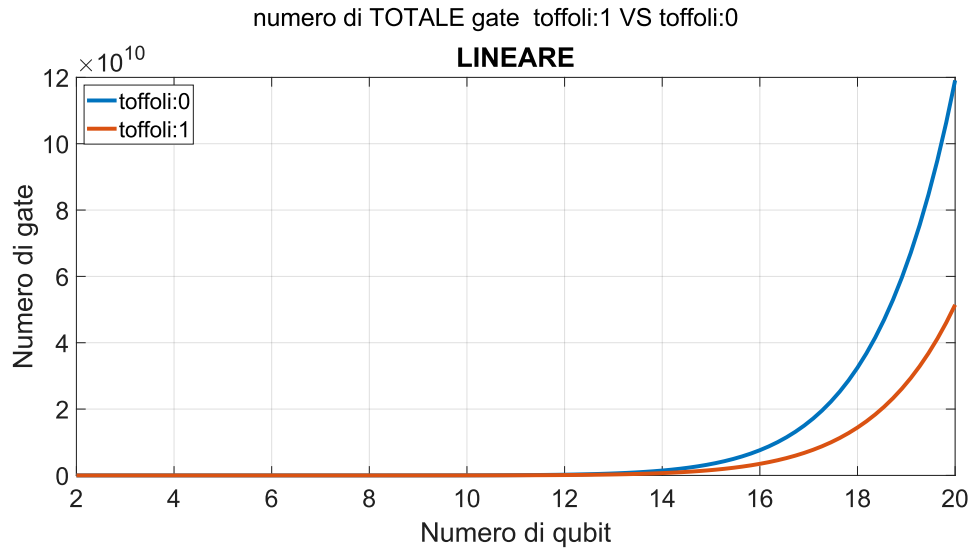


Figura 4.37: Grafico con ilo fitting dei dati del numero di porte quantistiche utilizzato per la descrizione del circuito in OpenQASM 2.0 in caso di abilitazione della scomposizione della porta Toffoli `(toffoli:0)` o meno `(toffoli:1)`, in scala lineare.

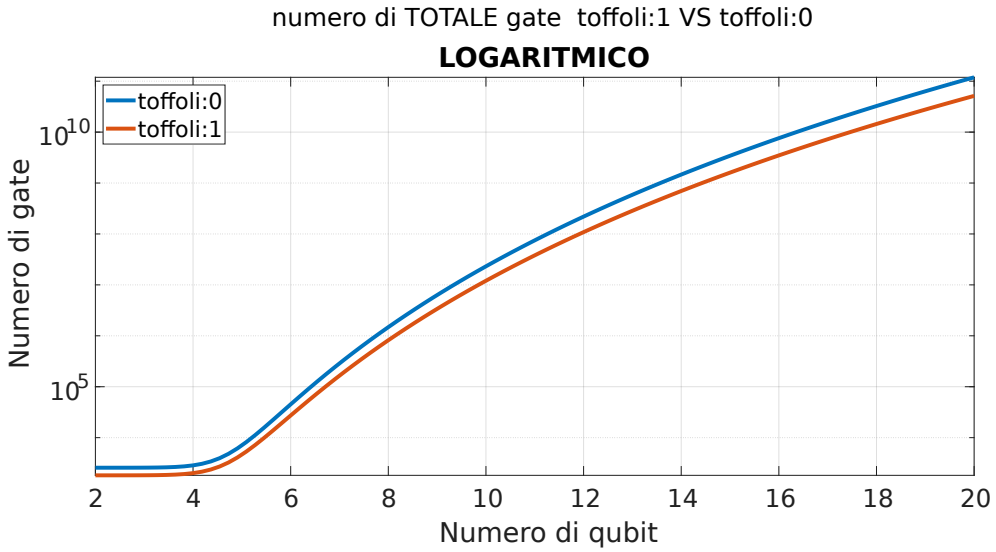


Figura 4.38: Grafico con ilo fitting dei dati del numero di porte quantistiche utilizzato per la descrizione del circuito in OpenQASM 2.0 in caso di abilitazione della scomposizione della porta Toffoli `(toffoli:0)` o meno `(toffoli:1)`, in scala semilogaritmica.

Numero di qubit	2	3	4	5	6	7	8
Tempo di scomposizione Li-Pelejo testEN:1 [s]	0.002	0.016	0.116	1.305	30.20	477.5	6305
Tempo di scomposizione Li-Pelejo testEN:0 [s]	0.002	0.010	0.054	0.324	4.089	43.45	604.0
Tempo di scomposizione Nielsen-Chuang toffoli:0 testEN:0 [s]	0	0.002	0.020	0.151	1.012	6.367	54.46
Tempo di scomposizione Nielsen-Chuang toffoli:0 testEN:1 [s]	0	0.004	0.030	0.208	1.369	8.204	64.91
Tempo di scomposizione Nielsen-Chuang toffoli:1 testEN:0 [s]	0	0.002	0.014	0.098	0.631	3.693	31.67
Tempo di scomposizione Nielsen-Chuang toffoli:1 testEN:1 [s]	0	0.003	0.023	0.014	0.864	4.828	37.55
Tempo di geenrazione OpenQASM 2.0 toffoli:0 testEN:0 [s]	0.002	0.006	0.035	0.219	1.257	6.776	34.16
Tempo di geenrazione OpenQASM 2.0 toffoli:0 testEN:1 [s]	0.002	0.007	0.042	0.262	1.520	8.115	42.05
Tempo di generazione OpenQASM 2.0 toffoli:1 testEN:0 [s]	0.002	0.006	0.029	0.157	0.835	4.207	20.34
Tempo di generazione OpenQASM 2.0 toffoli:1 testEN:1 [s]	0.002	0.007	0.036	0.189	1.000	5.031	24.87

Tabella 4.1: Tabella riassuntiva con tutti i risultati delle latenze di scomposizione dei diversi algoritmi adoperati.

Numero di qubit	2	3	4	5	6	7	8
Tempo di scomposizione Totale toffoli:0 testEN:0 [s]	0.003	0.018	0.109	0.694	6.358	56.60	692.6
Tempo di scomposizione Totale toffoli:0 testEN:1 [s]	0.004	0.027	0.187	1.775	33.09	493.9	6413
Tempo di scomposizione Totale toffoli:1 testEN:0 [s]	0.003	0.018	0.096	0.582	5.905	51.09	656.3
Tempo di scomposizione Totale toffoli:1 testEN:1 [s]	0.004	0.025	0.183	1.622	32.07	479.9	6260

Tabella 4.2: Tabella riassuntiva con tutti i risultati delle latenze totale di scomposizione.

Numero di qubit	2	3	4	5	6	7	8
Errore numerico Li-Pelejo	1.1e-15	1.8e-15	2.2e-15	4.4e-15	7.1e-15	1.4e-14	2.4e-14
Errore numerico Nielsen-Chuang toffoli:0	0	1.5e-15	2.3e-15	3.0e-15	3.5e-15	9.5e-15	1.1e-14
Errore numerico Nielsen-Chuang toffoli:1	0	1.7e-15	2.2e-15	2.6e-15	6.5e-15	8.4e-15	2.3e-14

Tabella 4.3: Tabella riassuntiva con tutti i risultati degli errori numerici.

Numero di qubit	2	3	4	5	6	7	8
Numero di porte quantistiche toffoli:0	22	188	1352	8764	51824	284676	1478120
Numero di porte quantistiche toffoli:1	22	188	1096	5980	31856	164804	825960

Tabella 4.4: Tabella riassuntiva con tutti i risultati del numero di porte adoperate per la descrizione del circuito in OpenQASM 2.0.

4.4 Metodo di verifica funzionale dell'algoritmo di scomposizione

In questa sezione è descritta la verifica funzionale del metodo di scomposizione delle matrici unitarie implementato in questa tesi.

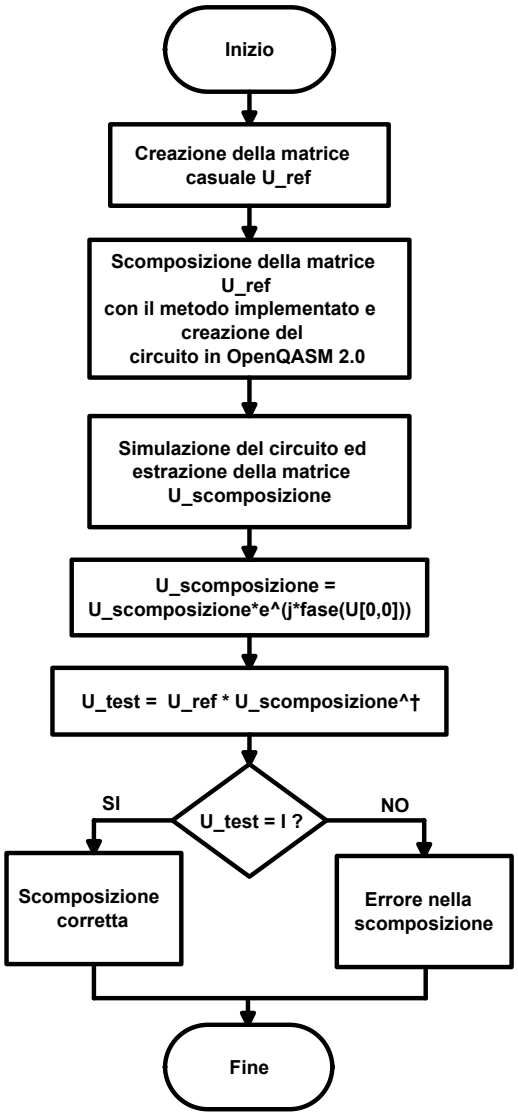


Figura 4.39: Diagramma di flusso della tecnica di verifica funzionale del metodo di scomposizione delle matrici unitarie implementato.

In Figura 4.39 è riportato il diagramma di flusso della tecnica di verifica. Il primo passo è quello di generare una matrice unitaria (U_{ref}) casuale su un numero di qubit deciso dall'utente e passato alla funzione come paramento di ingresso. La matrice U_{ref} viene scomposta dal metodo proposto di questa tesi generando un file in OpenQASM 2.0 contenente il circuito quantistico associato alla matrice unitaria. Il circuito quantistico viene simulato tramite `UnitarySimulator` di **Qiskit** e viene estratta la matrice associata al circuito $U_{scomposizione}$ tramite il metodo `get_unitary()`. Alla matrice $U_{\text{scomposizione}}$ viene moltiplicato il temine di fase globale della matrice U_{ref} ($U_{\text{scomposizione}} = U_{\text{scomposizione}} * e^{j*\angle U_{\text{ref}}[0,0]}$), che viene perso durante la scomposizione. Successivamente si calcola la matrice $U_{\text{test}} = U^{\dagger}_{\text{scomposizione}} U_{\text{ref}}$ ed infine si verifica che U_{test} sia uguale alla matrice identità, a meno di una tolleranza configurabile dall'utente.

Di seguito viene riportato un esempio dell'uscita del procedimento di verifica funzionale implementato.

```
(base) bash-4.2$ python3 test_scomposizione.py 2 2e-15
U_ref:

[[ 0.10668398+0.46533579j -0.17415801+0.58029768j
     -0.19376476+0.10563776j
   0.20051824-0.56222152j]
 [ 0.40596146+0.14662162j  0.48721157-0.05502495j
      0.03147499-0.43308569j
   0.6093723 +0.1157845j ]
 [-0.52856291+0.54583175j  0.46539318-0.38541339j
      -0.07851844+0.05254082j
  -0.10819762-0.19215165j]
 [ 0.01498609-0.09079043j  0.03904057-0.16081752j
      0.60039242+0.62717242j
   0.41708641-0.19070579j]]


fase globale di U_ref: 1.3454286349146987


```

```
Scomposizione della matrice U:

Creating file test_scomposizione.qasm...
OK

errors: 0

U_scomposizione:

[[ 0.47740849+0.j          0.52670494+0.29942999j
     0.05966679+0.21247116j
  -0.5031953 -0.32108424j]
 [ 0.23363193-0.36293074j  0.05524114-0.4871871j
      -0.41510027-0.12745845j
   0.24902978-0.56808874j]
 [ 0.41391357+0.63717078j -0.27166808-0.53975063j
      0.03366606+0.08827388j
  -0.21147087+0.0625224j ]
 [-0.08514566-0.03489558j -0.14802657-0.07399036j
      0.74547905-0.44505876j
  -0.0926791 -0.44915515j]]


U_scomposizione con la moltiplicazione della fase globale:

[[ 0.10668398+0.46533579j -0.17415801+0.58029768j
     -0.19376476+0.10563776j
   0.20051824-0.56222152j]
 [ 0.40596146+0.14662162j  0.48721157-0.05502495j
      0.03147499-0.43308569j
   0.6093723 +0.1157845j ]
 [-0.52856291+0.54583175j  0.46539318-0.38541339j
      -0.07851844+0.05254082j
  -0.10819762-0.19215165j]
 [ 0.01498609-0.09079043j  0.03904057-0.16081752j
      0.60039242+0.62717242j
   0.41708641-0.19070579j]]


```

```
48 U_test:

50 [[ 1.00000000e+00+8.32667268e-17j  6.93889390e-17+0.00000000e+00j
     5.55111512e-17-1.38777878e-16j  1.94289029e-16+1.38777878e-16j]
52  [-6.93889390e-17-1.11022302e-16j  1.00000000e+00+1.38777878e-16j
    -1.11022302e-16-4.16333634e-17j -5.55111512e-17+1.11022302e-16j]
54  [-8.32667268e-17+1.38777878e-17j  1.24900090e-16+1.11022302e-16j
     1.00000000e+00-1.11022302e-16j  2.77555756e-17-7.63278329e-17j]
56  [-1.80411242e-16+1.38777878e-16j -1.11022302e-16+1.66533454e-16j
    -1.94289029e-16+0.00000000e+00j  1.00000000e+00+1.66533454e-16j]]
58

60 tolleranza per il test : 2e-15

62 LA MATRICE U_test E LA MATRICE IDENTITA
   (base) bash-4.2$
```

Capitolo 5

Quantum Phase Estimation

In questo capitolo è riportata l'applicazione del metodo di scomposizione delle matrici unitarie per la realizzazione del circuito della Quantum Phase Estimation (QPE). La teoria alla base della QPE è riportata dettagliatamente nel libro [2], mentre in questa tesi è presentata solo una breve descrizione dell'algoritmo. Una descrizione approfondita della QPE esula, infatti, dagli obiettivi di questa tesi.

Data una matrice hermitiana H ed un autovettore unitario $|\psi\rangle$, si cerca di approssimare l'autovalore φ_ψ associato a $|\psi\rangle$ con la routine quantistica QPE. In particolare, costruita la matrice unitaria U associata alla matrice hermitiana H come:

$$U = e^{\frac{2\pi j}{2^n} \cdot H} \,, \tag{5.1}$$

dove n è il numero di *ancilla* qubit su cui rappresentare l'approssimazione dell'autovalore φ_ψ, è possibile dimostrare che applicando l'autovettore $|\psi\rangle$ alla matrice U si ottiene:

$$U\,|\psi\rangle = e^{j\varphi_\psi}\,|\psi\rangle \ . \tag{5.2}$$

Dove φ_ψ è un numero reale e corrispondente all'autovalore da ricercare.
Attraverso l'applicazione della QPE è possibile approssimare $e^{j\varphi_\psi}$ con $e^{j\frac{2\pi}{2^n}b}$, dove b è un numero binario nella forma $b_{n-1}b_{n-2}\cdots b_1 b_0$ codificato negli *ancilla* qubit.
Lo schema a blocchi del circuito della QPE è mostrato nella Figura 5.1. In particolare gli n qubit più in alto sono gli ancilla qubit su cui verrà salvato il valore

dell'approssimazione dell'autovalore alla fine dell'algoritmo, mentre, gli m qubit inferiori sono quelli su cui è codificato l'autovettore $|\psi\rangle$ dopo il blocco di *preparazione di stato*.

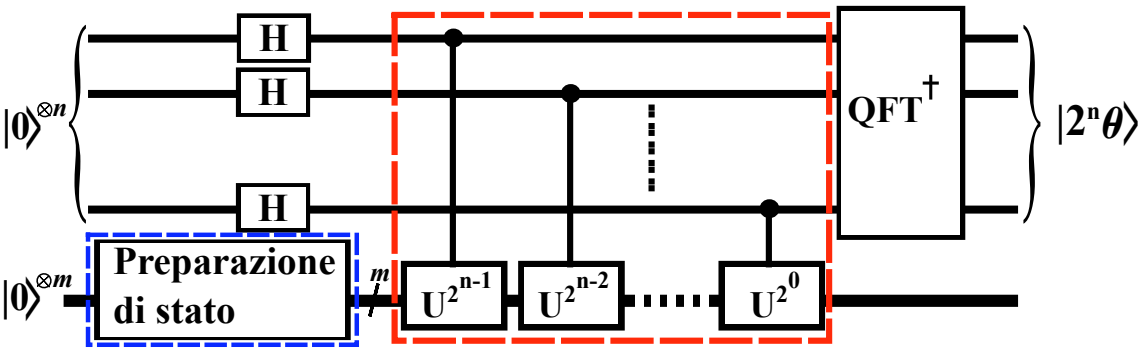


Figura 5.1: Schema a blocchi del circuito della Quantum Phase Estimation (QPE).

Lo schema della QPE può essere diviso in tre macro-blocchi logici:

1. **Preparazione dei qubit**, di cui fanno parte il blocco di preparazione di stato (che codifica il valore dell'autovettore $|\psi\rangle$ sugli m qubit più in basso) e le porte di Hadamard applicate agli ancilla qubit per portarli in una condizione di sovrapposizione di stati;

2. **Applicazione di rotazioni controllate di tipo** CU^{2^i}, (riquadro rosso nella Figura 5.1) queste rotazioni estraggono il valore dell'autovalore un qubit alla volta attraverso il *phase kick-back effect*[10], in cui l'effetto di una fase relativa viene spostato dal qubit target a quello di controllo.

3. **Applicazione dell' Inverse Quantum Fourier Transform** ($QFT^\dagger$): per convertite gli ancilla qubit dalla base di Fourier a quella computazionale.

Nello specifico, l'applicazione del metodo di scomposizione proposto in questa tesi è stato utilizzato per la realizzazione dei blocchi di preparazione di stato (blocco blu in Figura 5.1) e delle varie porte di rotazione controllate CU^{2^i} (blocco rosso in Figura 5.1). Per quanto concerne il blocco di preparazione di stato, sono stati implementati e confrontati due metodi per passare dal vettore unitario codificante l'autovettore $|\psi\rangle$ ad una matrice unitaria, necessaria per la scomposizione e realizzazione del circuito in OpenQASM 2.0. Invece, per la realizzazione delle varie porte

di rotazione controllate sono stati implementati e confrontati due metodologie di creazione delle varie porte CU^{2^i}: il primo prevede di eseguire una scomposizione diversa per ciascuna porta controllata, mentre l'altro stanzia 2^i volte il circuito della matrice CU^{2^0}, al fine di ottenere le rotazioni per le potenze maggiori di uno.

Nel prosieguo del capitolo sono riportati le implementazioni ed i risultati per tutti i metodi descritti, confrontando vantaggi e svantaggi di ognuno.

Per quanto concerne la realizzazione del circuito relativa alla $QFT^\dagger$ (Figura 5.2), è stato realizzato uno script Python specifico per la descrizione del circuito. Essendo la matrice associata alla $QFT^\dagger$ piena e considerando che: il tempo di scomposizione richiesto per scomporre una matrice di questo tipo è generalmente elevato, che la struttura del circuito è ben nota e facilmente costruibile per diversi parallelismi, risulta particolarmente futile utilizzare l'algoritmo di scomposizione proposto. Questa osservazione punta a precisare che l'algoritmo di scomposizione di matrici unitarie implementato è molto utile quando, data una matrice unitaria generica, non si conosce il circuito quantistico ad essa associata. In caso contrario conviene generare il circuito tramite uno script specifico.

I test funzionali sono stati effettuati partendo sempre da matrici Hermitiane (per semplicità reali simmetriche) generate casualmente, caratterizzate dalla presenza di un autovalore **intero**, che è lo scalare b da calcolare con la QPE. Il corrispettivo autovettore è stato utilizzato per generare il circuito di preparazione di stato, mentre la matrice Unitaria ottenuta dalla matrice Hemitiana secondo la formula 5.1 è adoperata per ottenere le evoluzioni unitarie associate alle rotazioni CU^{2^i}. Si verifica, infine, che la simulazione in Qiskit del suddetto circuito produca come risultato una probabilità pari a 1 di misurare gli n qubit superiori nello stato associato al valore binario di b. In questo contesto, è stato adoperato il simulatore compilato `QasmSimulator` di Qiskit, che a differenza dell'`UnitarySimulator` fornisce come risultato la distribuzione di probabilità. Alla fine di questo capitolo è riportato un esempio di verifica funzionale della QPE.

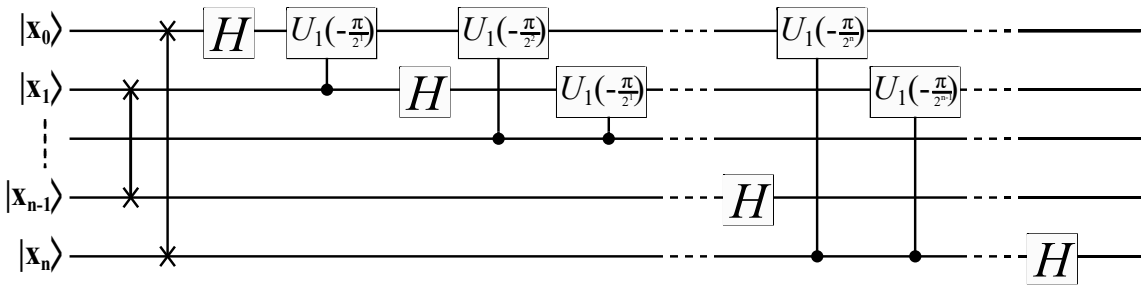


Figura 5.2: Circuito della $QFT^{\dagger}$.

5.1 Preparazione di stato

Per la realizzazione dei circuiti di preparazione di stato, è stato deciso di implementare due metodi per la creazione di una matrice unitaria partendo da un vettore unitario $|\psi\rangle$: il primo basato sulla trasformazione di Householder [11], l'altro consistente nella costruzione di una base ortonormale contenente $|\psi\rangle$ attraverso il calcolo del *null space* dello stesso $|\psi\rangle$ [12]. In particolare, con entrambi i metodi si ottinene una matrice unitaria avente $|\psi\rangle$ sulla colonna più a sinistra in modo tale che $|\psi\rangle = U\,|0\rangle^{\otimes n}$.
In questa sezione sono brevemente descritti i due algoritmi, quindi sono riportati i diagrammi di flusso utilizzati per la loro implementazione e i risultati delle varie scomposizioni.

5.1.1 Teoria

In questa sezione è spiegata la teoria alla base dei metodi di trasformazione di *Householder* e del *null space*.

Metodo trasformazione Householder La trasformazione di Householder[11], di seguito chiamato metodo di Householder, in uno spazio euclideo descrive una riflessione di un vettore rispetto ad un iperpiano contenente l'origine.
Dal punto di vista algebrico, considerando un vettore colonna unidimensionale w, la matrice unitaria $2D$ ad esso associata può essere calcolata come:

$$U = \mathbf{I} - 2\frac{w \cdot w^T}{w^T \cdot w}\,, \tag{5.3}$$

dove $\mathbf{I}$ è la matrice identità, w^T è il vettore w trasposto, il prodotto $w^T \cdot w$ è la norma della moltiplicazione $w \cdot w^T$.

Metodo di scomposizione del null space Il metodo di scomposizione del null space[12], di seguito chiamato metodo di Nullspace, si basa sul calcolo del nucleo o *kernel* dello spazio lineare in cui si trova il vettore di partenza. Dato un vettore colonna unitario $|\psi\rangle$ di lunghezza 2^n, sono calcolati i $2^n - 1$ vettori ad esso ortonormali $|\psi\rangle_{\text{ort.},k}$:

$$\langle\psi|\psi_{\text{ort.},k}\rangle = 0\,. \tag{5.4}$$

L'operazione può essere effettuata analiticamente o facendo ricorso a librerie software per il calcolo numerico. I vettori $|\psi\rangle$ e $|\psi\rangle_{\text{ort.},k}$ costituiscono una base ortonormale e la matrice ottenuta incolonnando questi vettori è unitaria:

$$U = \begin{bmatrix} |\psi\rangle & |\psi\rangle_{\text{ort.},1} & \cdots & |\psi\rangle_{\text{ort.},2^n-1} \end{bmatrix}\,. \tag{5.5}$$

La dimostrazione è di seguito riportata. Sapendo per definizione che $U^\dagger U = \mathbf{I}$, dove

$$U^\dagger = \begin{bmatrix} \langle\psi| \\ \langle\psi_{\text{ort.},1}| \\ \vdots \\ \langle\psi|_{\text{ort.},2^n-1} \end{bmatrix}\,, \tag{5.6}$$

si ottiene che

$$U^\dagger U = \begin{bmatrix} \langle\psi| \\ \langle\psi_{\text{ort.},1}| \\ \vdots \\ \langle\psi|_{\text{ort.},2^n-1} \end{bmatrix} \begin{bmatrix} |\psi\rangle & |\psi\rangle_{\text{ort.},1} & \cdots & |\psi\rangle_{\text{ort.},2^n-1} \end{bmatrix} = \\ = \begin{bmatrix} \langle\psi|\psi\rangle & \langle\psi|\psi_{\text{ort.},1}\rangle & \cdots & \langle\psi|\psi_{\text{ort.},2^n-1}\rangle \\ \langle\psi_{\text{ort.},1}|\psi\rangle & \langle\psi_{\text{ort.},1}|\psi_{\text{ort.},1}\rangle & \cdots & \langle\psi_{\text{ort.},1}|\psi_{\text{ort.},2^n-1}\rangle \\ \vdots & \ddots & \ddots & \vdots \\ \langle\psi_{\text{ort.},2^n-1}|\psi\rangle & \langle\psi_{\text{ort.},2^n-1}|\psi_{\text{ort.},1}\rangle & \cdots & \langle\psi_{\text{ort.},2^n-1}|\psi_{\text{ort.},2^n-1}\rangle \end{bmatrix}. \tag{5.7}$$

Sapendo che i vettori costituiscono una base ortonormale, i prodotti scalari associati agli elementi sulla diagonale principale sono pari a 1, mentre quelli dei termini fuori-diagonale sono pari a 0. La matrice risultante è, dunque, una matrice identità, come volevasi dimostrare.

5.1.2 Implementazione

In questa sezione sono riportati i diagrammi di flusso per l'implementazione dei due metodi di creazione di matrici unitarie partendo da vettori unitari.

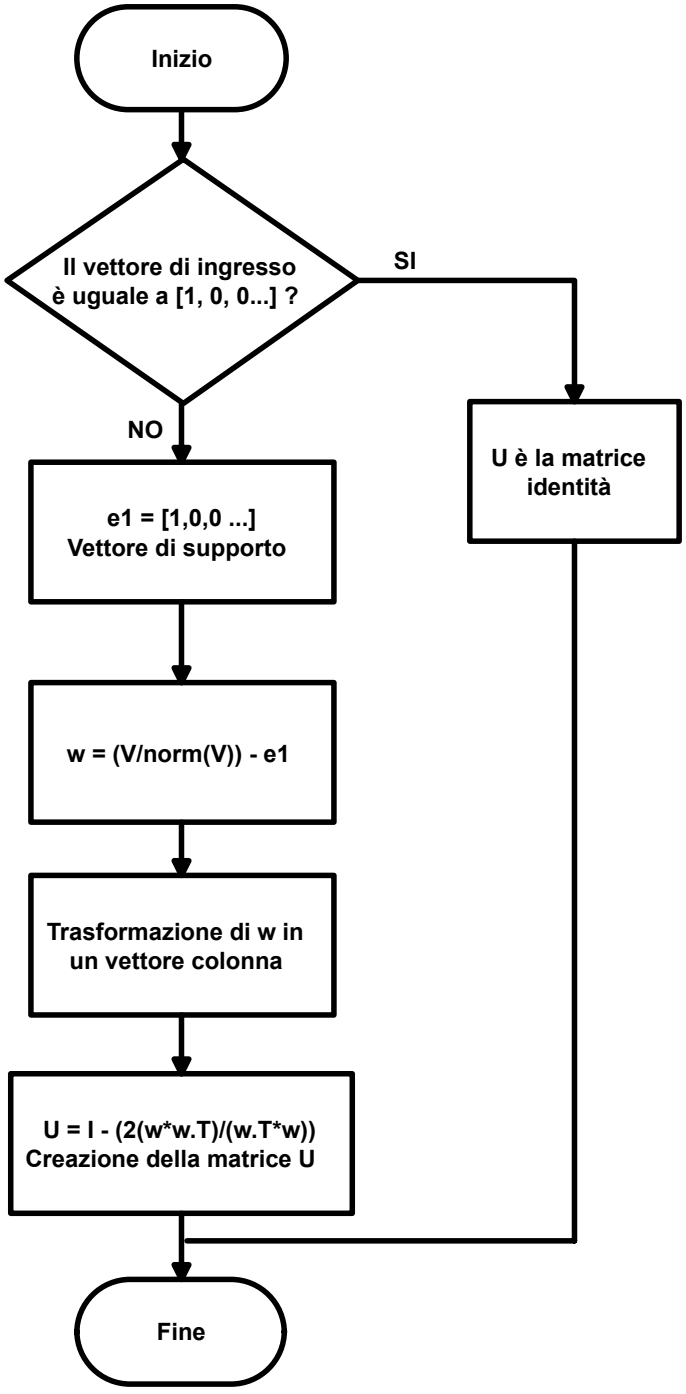


Figura 5.3: Diagramma di flusso per l'implementazione del metodo di Householder per la creazione di matrici unitarie partendo da vettori unitari.

Metodo Householder Il diagramma di flusso per l'implementazione del metodo di Householder è riportato in Figura 5.3. La funzione prende come parametro di ingresso una lista v passata per semplicità come vettore riga. Se tale lista è uguale a [1, 0, 0, ...], allora la matrice unitaria risultante è la matrice identità. Escluso questo particolare caso, l'algoritmo crea un vettore di supporto $e1$ uguale ad [1, 0, 0, ...] e lo usa per il calcolo del vettore w come $w = \frac{v}{\|v\|} - e1$. Il secondo passo consiste nel trasformare il vettore w in un vettore colonna ed utilizzarlo per la creazione della matrice U secondo la formula $U = \mathbf{I} - 2\frac{w \cdot w.T}{w.T \cdot w}$. Dove $w.T$ è il vettore w trasposto.

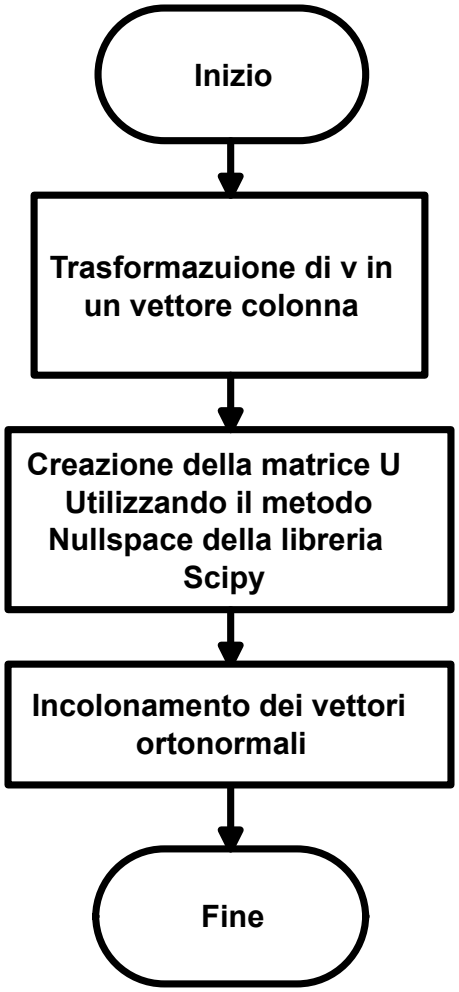


Figura 5.4: Diagramma di flusso per l'implementazione del metodo Nullspace per la creazione di matrici unitarie partendo da vettori unitari.

Metodo del Nullspace Il diagramma di flusso dell'implementazione del metodo di Nullspace è riportato nella Figura 5.4. La funzione riceve come parametro di ingresso una lista v rappresentante un vettore riga. Il primo passo dell'algoritmo consiste nel trasformare il vettore riga in un vettore colonna. Successivamente viene applicato il metodo Nullspace sfruttando il corrispettivo metodo contenuto all'interno della libreria *Scipy*. Infine si incolonnano i vettori cosi ottenuti, a destra di quello in ingresso, in modo tale che questo sia sulla colonna più a sinistra associata all'evoluzione di $|0\rangle^{\otimes n}$.

5.1.3 Risultati

In questa sotto-sezione sono riportati i risultati, in termini di latenza totale e numero di porte utilizzato per descrizione del circuito in OpenQASM 2.0, delle varie scomposizioni con i due metodi implementati per la costruzione di matrici unitarie partendo da vettori unitari.
È importante sottolineare che i risultati riportati nei grafici, sono ottenuti mediando dieci scomposizioni per ogni valore del numero di qubit, adoperando, a parità di numero di qubit, dieci vettori unitari diversi tra loro. Per ciascun vettore unitario sono state generate (e successivamente scomposte) le matrici unitarie con entrambi i metodi, garantendo così il confronto dei due metodi con gli stessi dati di ingresso.

Prima di mostrare i risultati per le scomposizioni con i due metodi, è riportato un esempio di un vettore unitario e le due matrici generate.
Il vettore unitario è:

$$v = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}, \tag{5.8}$$

la matrice generata con il metodo di Householder è:

$$U1 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.9}$$

mentre la matrice generata con il metodo di Nullspace è:

$$U2 = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{2} & \frac{5}{6} & -0.2357 & 0 \\ \frac{1}{\sqrt{2}} & -0.2357 & \frac{2}{3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.10}$$

Le due matrici sono simili ma non identiche. Questo esempio dimostra che con i due metodi si ottengono matrici diverse. In alcune casi le matrici potrebbero divergere anche nel numero di elementi nulli presenti all'interno di esse.

Nelle Figure 5.5 e 5.6 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del tempo totale di scomposizione, in caso di test disabilitati, delle matrici unitarie generate con i metodi di Householder e Nullspace. Nelle Figure 5.7 e 5.8 sono riportati i grafici corrispettivi con i test abilitati. In tutti e quattro i grafici non è possibile notare grandi differenze in termini di latenza di scomposizione tra questi due algoritmi. È interessante notare che in entrambi i casi, gli algoritmi presentano tempi di scomposizione leggermente inferiori rispetto a quelli di matrici generiche riportate nel capitolo 4. Questo si può giustificare con la presenza di alcuni 0 all'interno delle matrici generate, che permettono una lieve riduzione della latenza dell'algoritmo di scomposizione di Li-Pelejo.

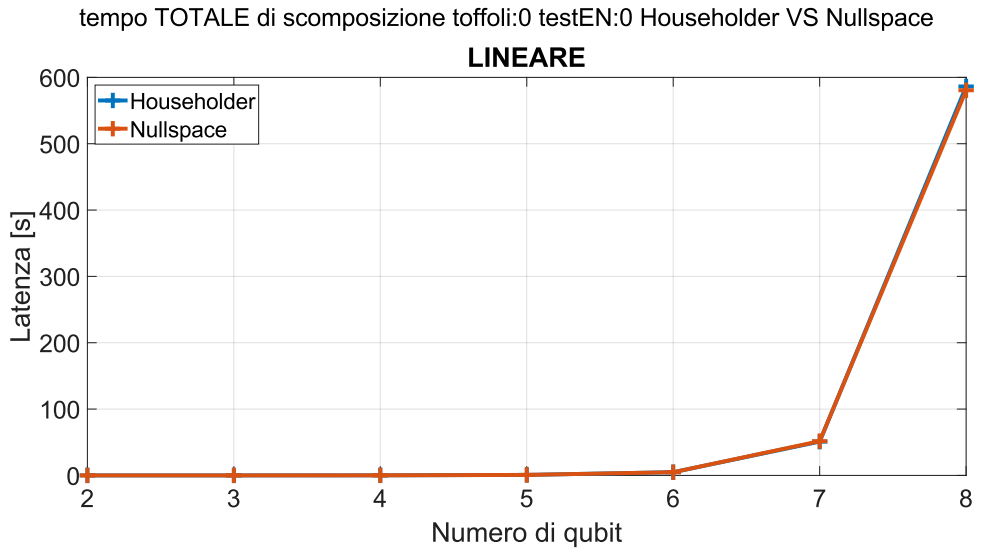


Figura 5.5: Grafico con il tempo di scomposizione, con test disabilitati, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala lineare.

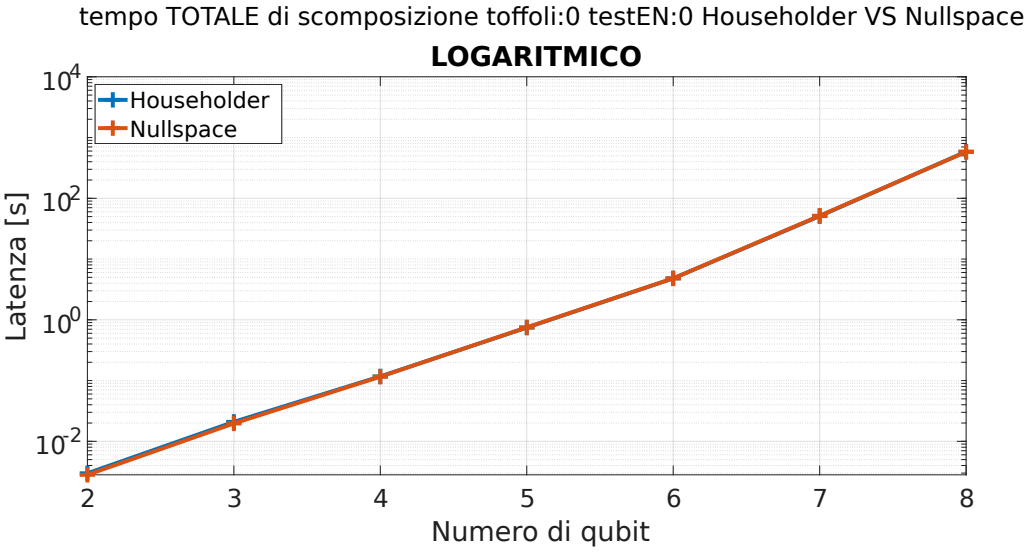


Figura 5.6: Grafico con il tempo di scomposizione, con test disabilitati, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala semi-logaritmica.

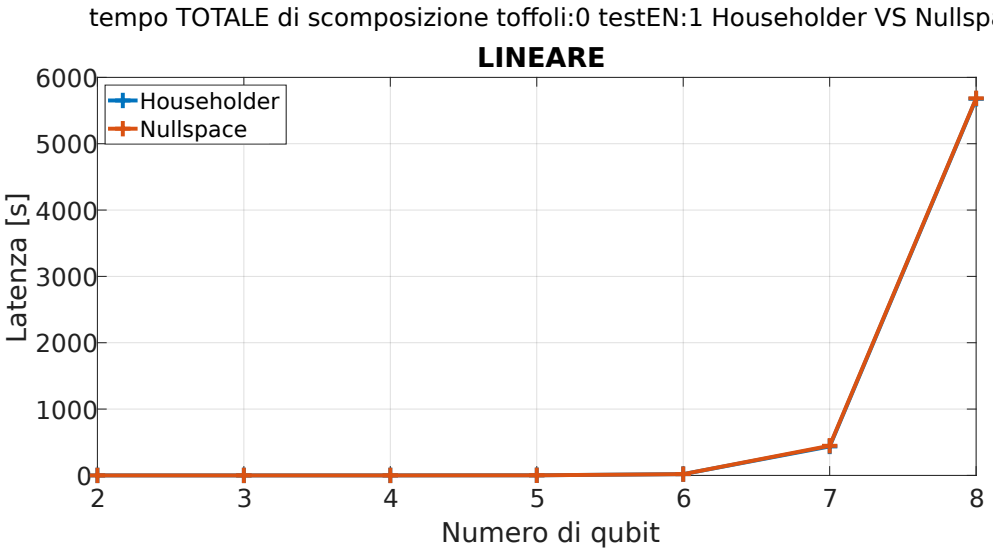


Figura 5.7: Grafico con il tempo di scomposizione, con test abilitati, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala lineare.

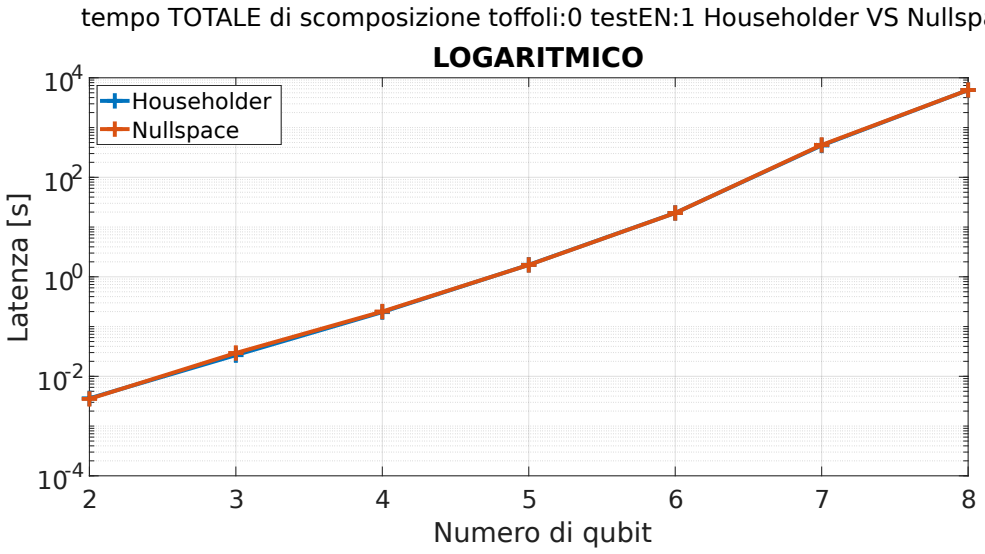


Figura 5.8: Grafico con il tempo di scomposizione, con test abilitati, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala semi-logaritmica.

Nelle Figure 5.9 e 5.10 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del numero totale di porte quantistiche adoperate per la descrizione del circuito in OpenQASM 2.0, in caso di disabilitazione della scomposizione della porta di Toffoli, delle matrici unitarie generate con i metodi Householder e Nullspace, mentre nelle Figure 5.11 e 5.12 sono riportati i grafici corrispettivi, in caso di abilitazione della scomposizione della porta di Toffoli. In tutti e quattro i grafici non è possibile notare differenza significative nel numero di porte utilizzate tra i due metodi implementati.

Alla luce dei risultati ottenuti riguardanti il tempo di scomposizione e il numero di porte quantistiche utilizzate, si può concludere che non vi sono differenze significative, in termini di prestazioni, tra le due metodologie di costruzione di matrici unitarie esaminate.

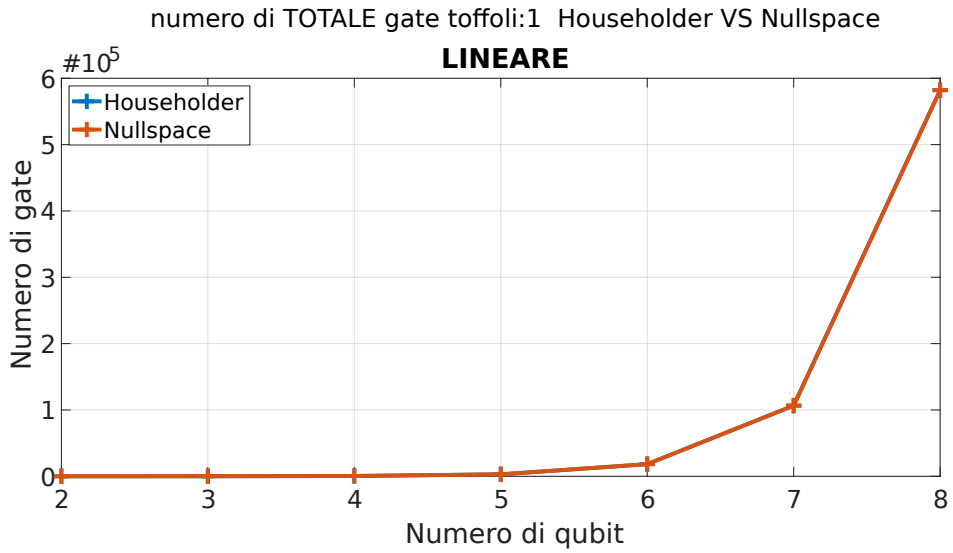


Figura 5.9: Grafico con il numero di porte quantistiche, con disabilitazione della scomposizione della porta Toffoli, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala lineare.

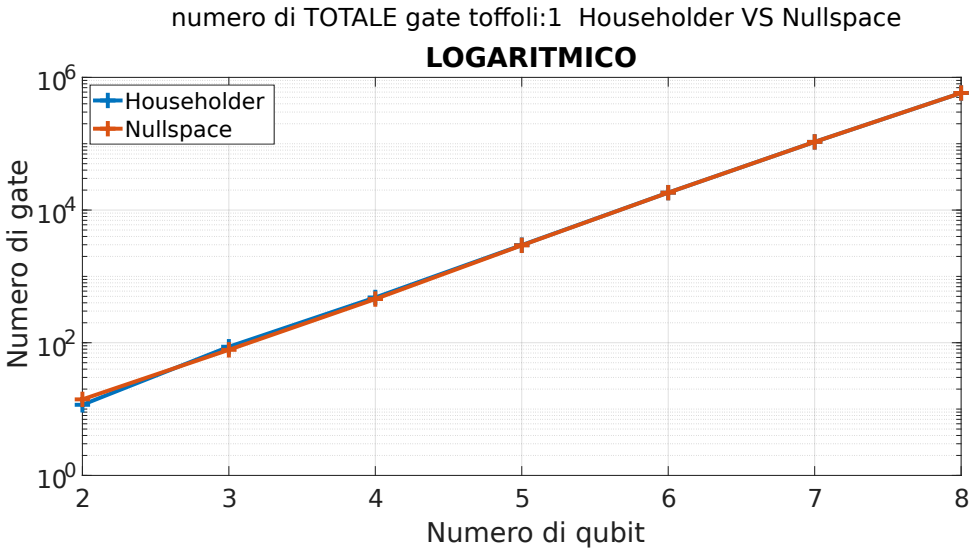


Figura 5.10: Grafico con il numero di porte quantistiche, con disabilitazione della scomposizione della porta Toffoli, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala semi-logaritmica.

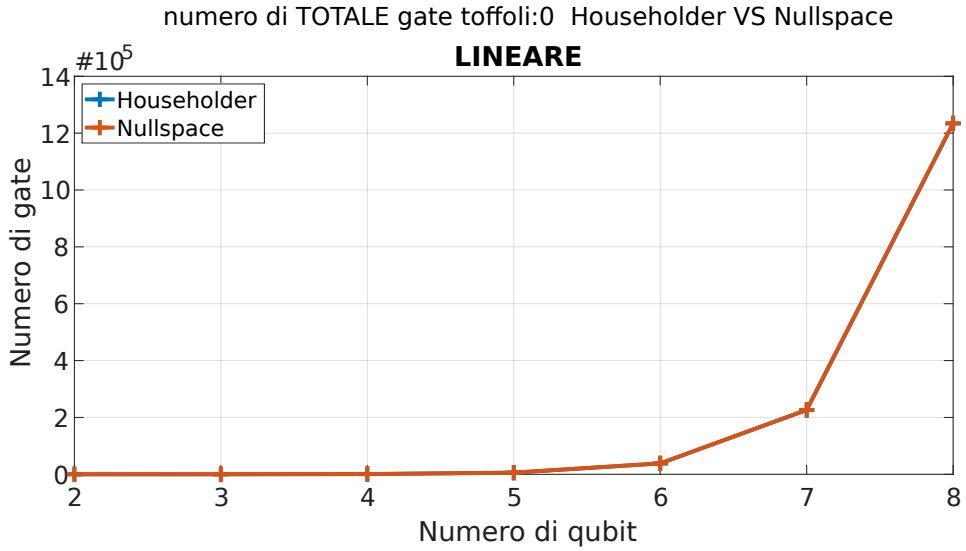


Figura 5.11: Grafico con il numero di porte quantistiche, con abilitazione della scomposizione della porta Toffoli, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala lineare.

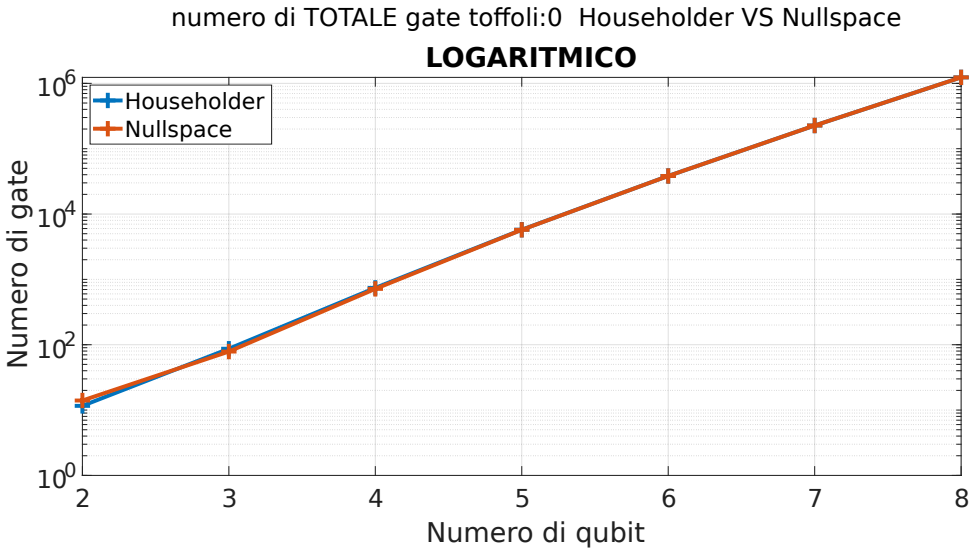


Figura 5.12: Grafico con il numero di porte quantistiche, con abilitazione della scomposizione della porta Toffoli, delle matrici unitarie generate con i metodi di Householder e Nullspace, in scala semi-logaritmica.

5.2 Rotazioni controllate CU^{2^i}

In questa sezione sono spiegati i due metodi di creazione delle porte controllate CU^{2^i} implementati: il primo prevede di eseguire una scomposizione diversa per ciascuna porta controllata, per cui è effettuata una scomposizione per la porta CU^{2^0}, dopodiché è eseguita la moltiplicazione matriciale $U \cdot U$ per ottenerne la matrice U^{2^1} ed è effetuata la scomposizione per la matrice CU^{2^1}, si procede egualmente fino alla potenza 2^{n-1}.
L'altra metodologia, invece, stanzia 2^i volte il circuito della matrice CU^{2^0}, al fine di ottenere le rotazioni per le potenze maggiori di uno, per cui è stanziato una volta il circuito per la rotazione controllata CU^{2^0}, due volte il circuito per la matrice CU^{2^1}, quattro volte per la matrice CU^{2^2} e cosi via fino alla potenza 2^{n-1}.

5.2.1 Implementazione

In questa sotto-sezione sono riportati i diagrammi di flusso per l'implementazione delle due metodologie implementate.

Metodologia multi-scomposizione Nella Figura 5.13 è riportato il diagramma di flusso nel caso in cui si decida di eseguire più scomposizione per le diverse potenze della matrice CU (metodo a multi-scomposizione). In questo caso è necessario un solo indice i per tenere conto della potenza della matrice CU appena scomposta.

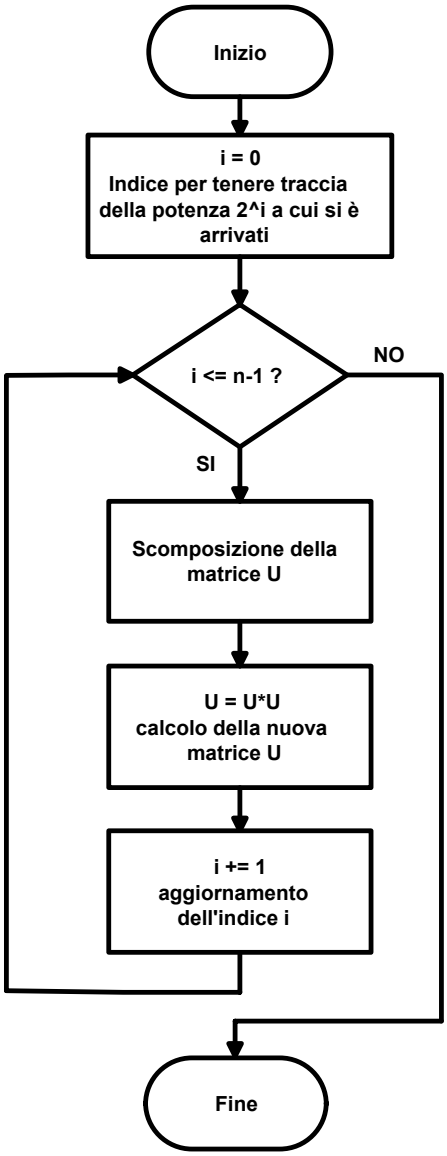


Figura 5.13: Diagramma di flusso nel caso in cui si decida di eseguire più scomposizione per le diverse potenze della matrice CU.

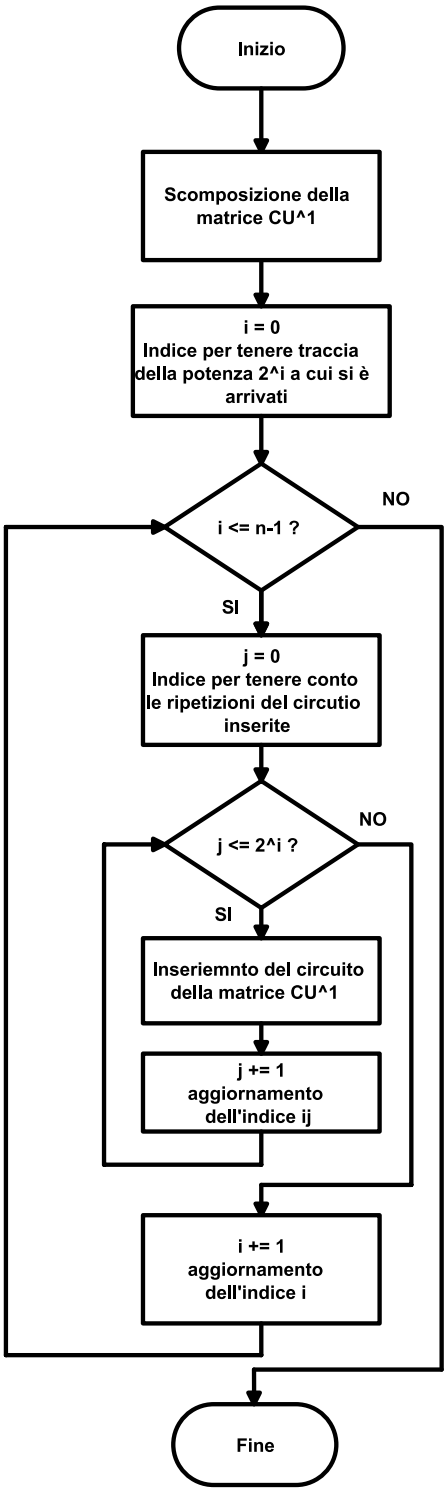


Figura 5.14: Diagramma di flusso nel caso in cui si decida di stanziare più volte il circuito relativo alla matrice CU^1 per le potenze maggiori di 1 della matrice CU^{2^i}.

Metodologia singola-scomposizione Nella Figura 5.14 è riportato il diagramma di flusso nel caso in cui si decida di stanziare più volte il circuito relativo alla matrice CU^1 per ottenere le matrici CU^{2^i} (metodo a singola-scomposizione). In questo caso sono necessari due indici: l'indice i per tenere conto della potenza della matrice CU^{2^i} e l'indice j per contare il numero di volte che è stato stanziato il circuito della CU^1 per ogni potenza 2^i.

5.2.2 Risultati

In questa sotto-sezione sono riportati i risultati relativi al tempo totale di generazione e numero di porte quantistiche adoperate nella costruzione del circuito di rotazione rappresentato dal blocco rosso in Figura 5.1 (chiamato di seguito CU_{rotation}). Si vuole precisare che risultati riportati nei vari grafici si riferiscono ai tempi di tutto il blocco rosso, non alle singole porte CU^{2^i}.
È importante evidenziare che tutti i risultati riportati nei grafici, sono stati ottenuti con $m = 4$ e $2 \leq n \leq 8$ (si veda la Figura 5.1 per chiarimenti) e sono valori medi di dieci scomposizioni, a parità di n.

Nelle Figure 5.15 e 5.16 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del tempo totale di costruzione del blocco CU_{rotation}, con test disabilitati, nei casi di multi-scomposizione e di singola-scomposizione, mentre nelle Figure 5.17 e 5.18 sono riportati i grafici corrispettivi con di test abilitati.
In tutti e quattro i grafici è possibile osservare che, nel caso di scomposizione singola, il tempo necessario alla creazione del blocco CU_{rotation} è circa costante. Invece, per quanto riguarda la scomposizione multipla, la curva presenta una crescita lineare all'aumentare del numero di ancilla qubit. Questi due andamenti sono concordi con quanto ci si aspetterebbe dalla teoria, in quanto, nel caso di scomposizione singola, il tempo di scomposizione della porta CU^{2^0} è dominante rispetto al tempo di scrittura del OpenQASM 2.0, per cui lo stanziamento ripetuto del circuito della CU^{2^0} per costruire le porte CU^{2^i} con potenza superiore a 1 non comporta dei significativi aumenti di latenza. Al contempo, nel caso di scomposizione multipla, l'aumento di latenza è proporzionale al numero di scomposizioni eseguite.

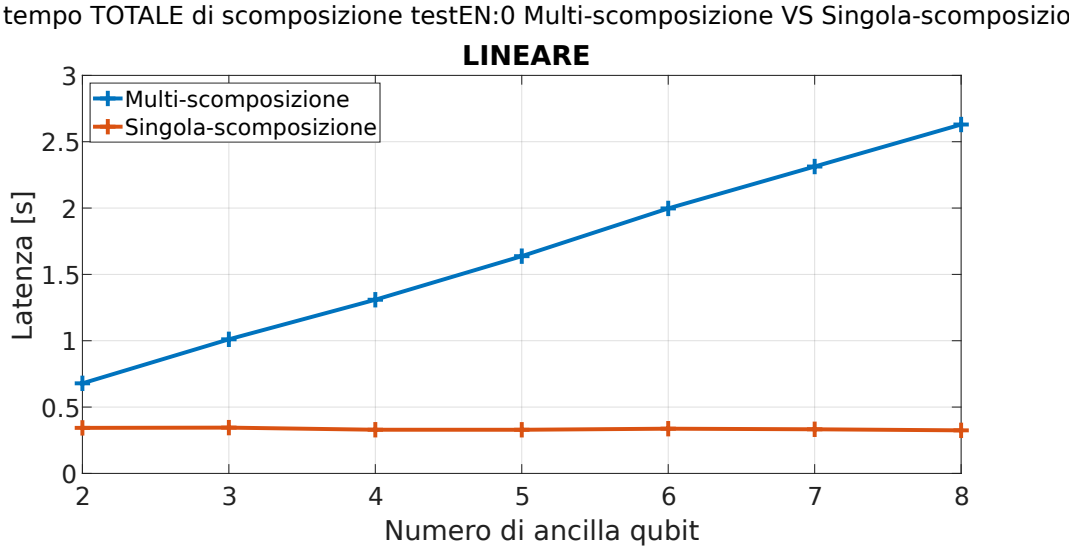


Figura 5.15: Grafico con il tempo di creazione del blocco CU_{rotation}, per test disabilitati, nei casi di multi-scomposizione e di singola-scomposizione, in scala lineare.

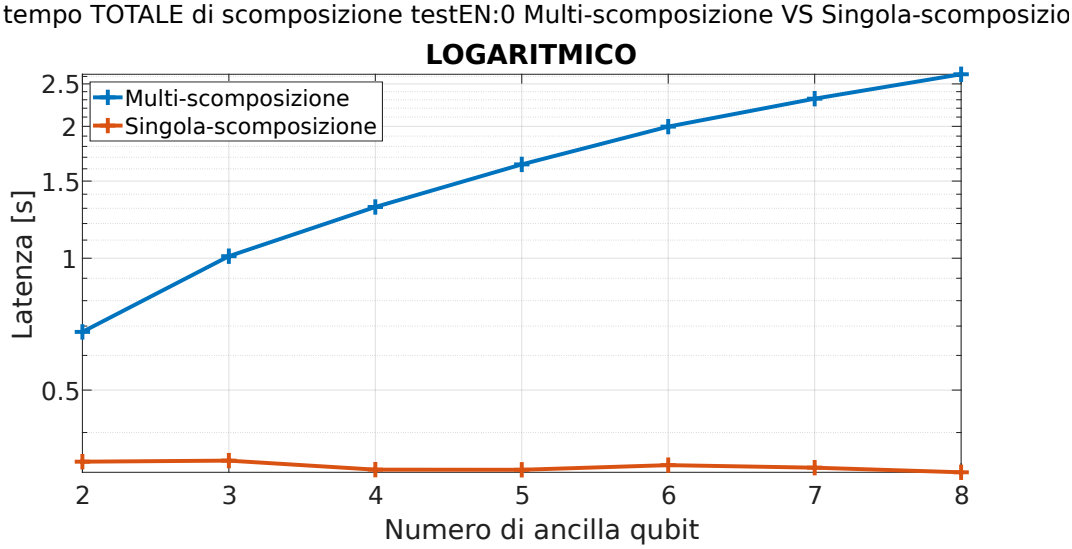


Figura 5.16: Grafico con il tempo di creazione del blocco CU_{rotation}, per test disabilitati, nei casi di multi-scomposizione e di singola-scomposizione, in scala semi-logaritmica.

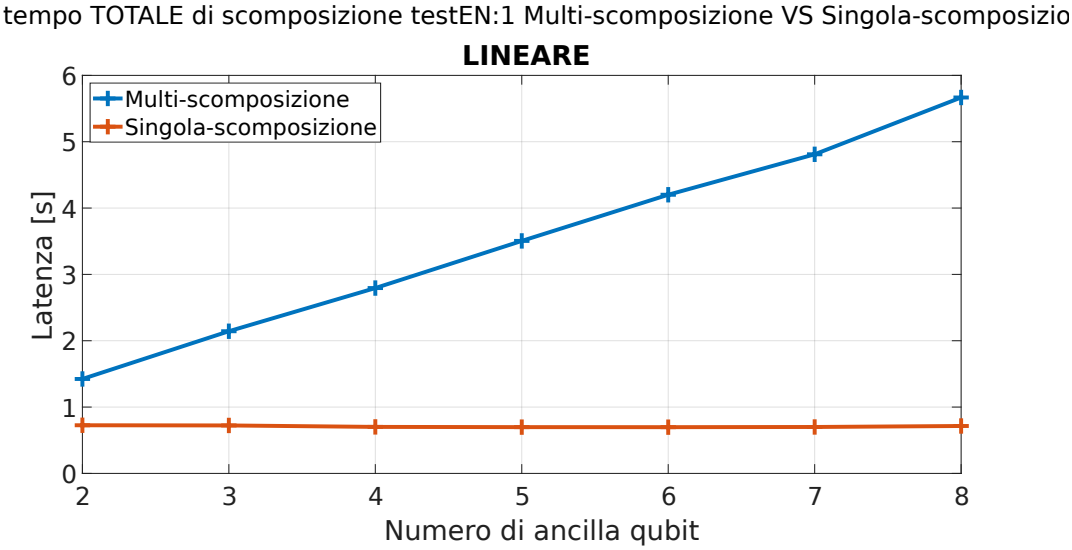


Figura 5.17: Grafico con il tempo di creazione del blocco CU_{rotation}, per test abilitati, nei casi di multi-scomposizione e di singola-scomposizione, in scala lineare.

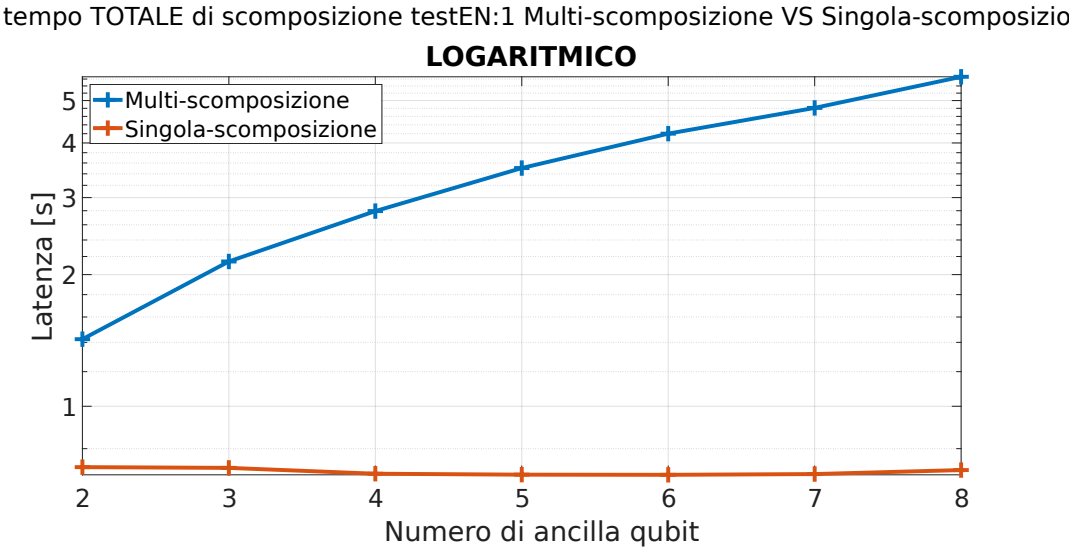


Figura 5.18: Grafico con il tempo di creazione del blocco CU_{rotation}, per test abilitati, nei casi di multi-scomposizione e di singola-scomposizione, in scala semi-logaritmica.

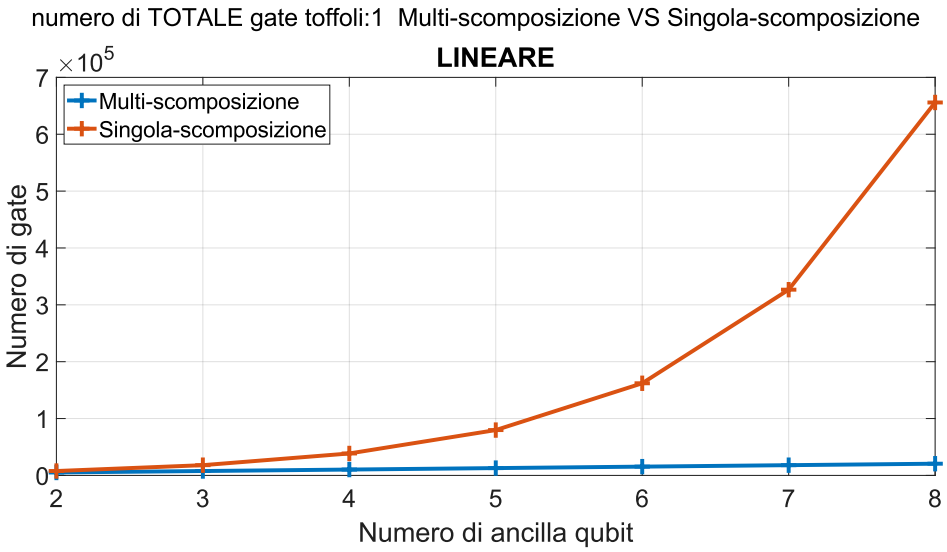


Figura 5.19: Grafico con il numero di porte totali del blocco CU_{rotation}, per la scomposizione della porta Toffoli disabilitata, nei casi di multi-scomposizione e di singola-scomposizione, in scala lineare.

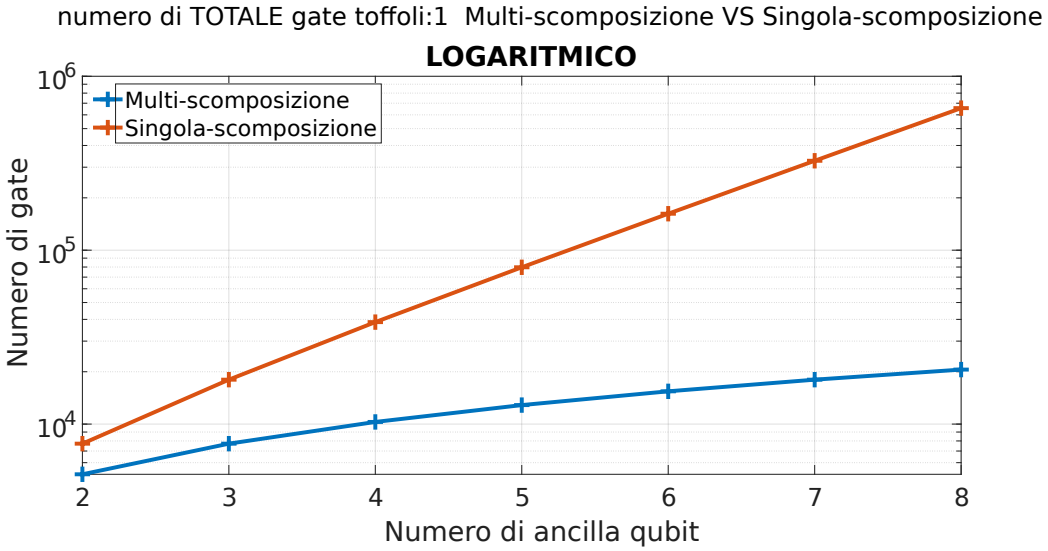


Figura 5.20: Grafico con il numero di porte totali del blocco CU_{rotation}, per la scomposizione della porta Toffoli disabilitata, nei casi di multi-scomposizione e di singola-scomposizione, in scala semi-logaritmica.

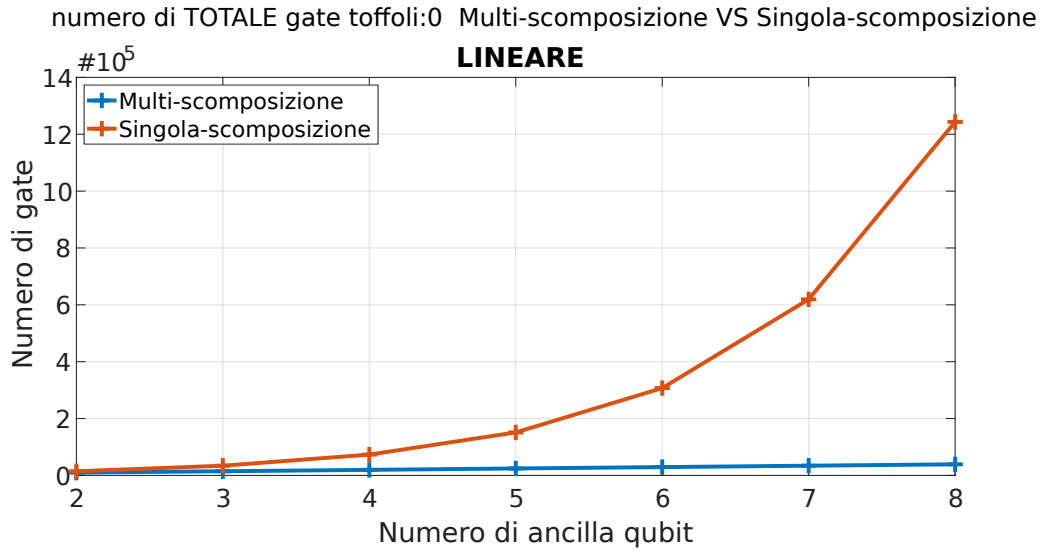


Figura 5.21: Grafico con il numero di porte totali del blocco CU_{rotation}, per la scomposizione della porta Toffoli abilitata, nei casi di multi-scomposizione e di singola-scomposizione, in scala lineare.

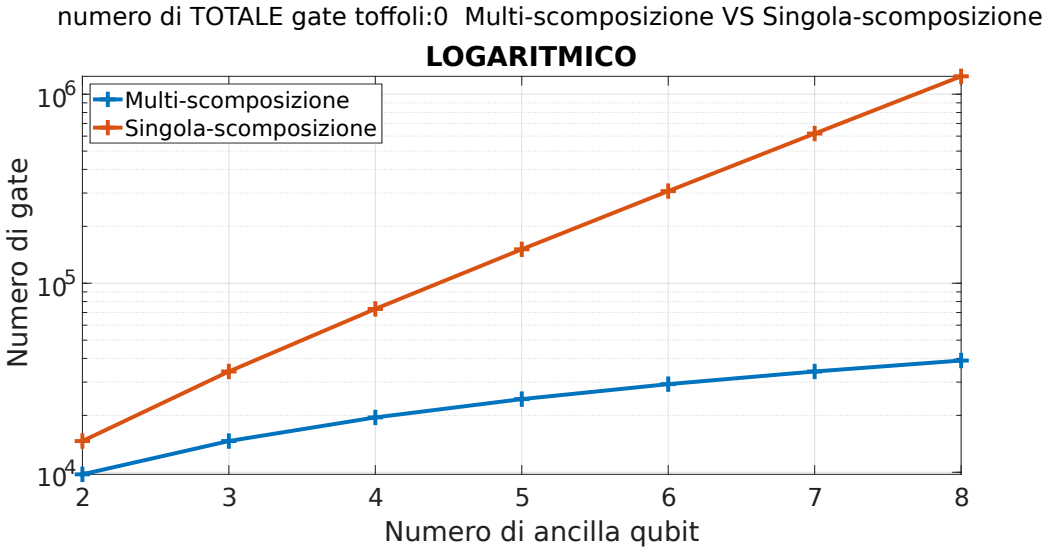


Figura 5.22: Grafico con il numero di porte totali del blocco CU_{rotation}, per la scomposizione della porta Toffoli abilitata, nei casi di multi-scomposizione e di singola-scomposizione, in scala semi-logaritmica.

Nelle Figure 5.19 e 5.20 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del numero di porte quantistiche totali adoperate per la descrizione in OpenQASM 2.0 del blocco CU_{rotation}, con la scomposizione della porta Toffoli disabilitata, nei casi di multi-scomposizione e di singola-scomposizione, mentre nelle Figure 5.21 e 5.22 sono riportati i grafici corrispettivi con la scomposizione della porta Toffoli abilitata.
In tutti e quattro i grafici è possibile osservare che nel caso di scomposizione multipla, il numero di porte utilizzato cresce linearmente (fenomeno particolarmente evidente nei grafici con asse semi-logaritmico 5.20 e 5.22, dove la curva assume un andamento $\propto \log(x)$), perché al crescere della potenza 2^i il numero di porte utilizzato per la descrizione della porta CU^{2^i} è lo stesso di quello della porta CU^{2^0}. Invece, nel caso di scomposizione singola il numero di porte utilizzato per la descrizione del blocco CU_{rotation} cresce esponenzialmente all'aumentare del numero di ancilla qubit, perché per la descrizione della porta CU^{2^i} sono necessarie il doppio delle porte usate per la descrizione di $CU^{2^{i-1}}$.
Nota importante, nei grafici 5.19 e 5.21 sembrerebbe che la curva nei casi di multi-scomposizione sia costante, in realtà è un problema di scala dovuto all'andamento esponenziale nei casi di singola-scomposizione.

In conclusione, la scomposizione singola ha una latenza poco più alta della una singola scomposizione di una porta CU^1, ma richiede un numero di porte che cresce esponenzialmente all'aumentare del numero di ancilla qubit. Invece, la scomposizione multipla richiede una tempo che cresce linearmente all'aumentare del numero di ancilla qubit e anche il numero di porte utilizzate per la descrizione del blocco CU_{rotation} cresce linearmente con il numero di ancilla qubit.

5.3 Metodo di verifica funzionale dell'implementazione della QPE

In questa sezione si riporta la metodologia per la verifica funzionale dell'algoritmo Quantum Phase Estimation.

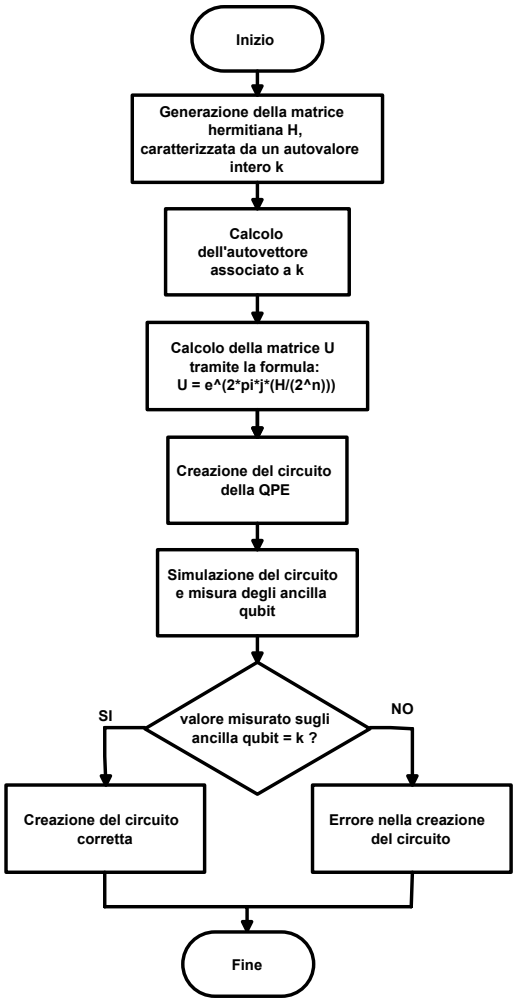


Figura 5.23: Diagramma di flusso della tecnica di verifica funzionale dell'implementazione della QPE.

Nella Figura 5.23 è riportato il diagramma di flusso della tecnica di verifica funzionale per la QPE. Il primo passo consiste nella generazione di una matrice Hermitiana H con almeno un autovalore k intero. Successivamente viene calcolato l'autovettore associato a k e la matrice unitaria U associata ad H, secondo l'Equazione (5.1). Si crea il circuito della QPE (garantendo che gli *ancilla qubit* siano di numero sufficiente a rappresentare il valore binario di k) e si simula utilizzando il `QasmSimulator`. Si misura lo stato degli *ancilla qubit* e si verifica che la probabilità di misurare k sia unitaria. Se questa condizione è soddisfatta, la creazione del circuito della QPE può essere ritenuta corretta.

Di seguito viene riportato un esempio del procedimento di verifica funzionale per la QPE implementato. Nel caso riportato, come si nota nella penultima riga, sono state effettuate 8192 misure per la costruzione della distribuzione di probabilità degli *ancilla qubit* e che tutte riportano il risultato previsto.

```
(base) bash-4.2$ python3 testQPE.py
matrice hermitiana H:

[[ 6  5  7 10]
 [ 5 10  7  8]
 [ 7  7  8  3]
 [10  8  3  0]]


autovalore: 3  numero di ancilla qubit n: 3


autovettore: [0.21320071635560955, 0.21320071635561127,
     -0.8528028654224419, 0.4264014327112207]


matrice unitaria U:

[(-0.2351592652952010 4+0.31218038145534405j),
     (0.4287892768904141+0.017932454900859315j),
     (-0.05700167533314998+0.1831319948397072j),
     (-0.5643717470571804+0.5547609620945858j)]
[(0.42878927689041413+0.017932454900859575j),
     (-0.5640319827447188+0.5595926579221534j),
     (0.04832797658077652+0.1446952502254203j),
     (-0.18927608450456837+0.3541813346326085j)]
[(-0.057001675333149926+0.18313199483970669j),
     (0.04832797658077664+0.14469525022542082j),
     (-0.551068611521039+0.7184863825328177j),
     (0.3164131887072043-0.141154419840023 03j)]
```

```
[(-0.5643717470571805+0.5547609620945859j),
     (-0.18927608450456831+0.35418133463260815j),
     (0.31641318870720436-0.14115441984002303j),
     (0.30254351200873486-0.029673206857095735j)]

Creating file qpe.qasm...
OK

Creation of the file qpe.inc...
OK


INVERSE QFT circuit generation...
Creating file inverse_qft_3.qasm...
OK

OK


U rotation circuit generation...
Creating file urotation.qasm...
OK

Creation of the file urotation.inc...
OK


U decompose circuit generation...
  Generation matrix U^0...
  OK

OK


State preparation circuit generation...
Creating file state_preparation.qasm...
OK

```

```
OK

Risultato atteso:
autovalore:3.0 ---> 11
Risultato simulazione:
{'011': 8192}
(base) bash-4.2$
```

Capitolo 6

Evoluzioni unitarie associate a look-up table

In questo capitolo sono illustrate alcune evoluzioni unitarie associate a look-up table. Con questa espressione si vogliono indicare tutti quegli operatori matematici, le cui matrici hanno un solo 1 su ogni riga ed ogni colonna, mentre tutti gli altri valori sono pari a 0. Un esempio di matrice con queste caratteristiche è la seguente:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} . \tag{6.1}$$

Che può essere interpretata come una look-up table in quanto l'indice delle colonne può essere visto come valore di ingresso al circuito, e l'indice di riga dell'elemento non nullo, su una colonna, indica il corrispondente valore dell'uscita.
La peculiarità di queste matrici è che sono matrici sparse, cioè presentano una grande quantità di elementi nulli. Questa caratteristica si traduce in una riduzione della latenza di scomposizione, che si può particolarmente apprezzare nell'algoritmo di scomposizione di Li-Pelejo. Infatti, dal momento che gran parte degli elementi fuori dalla diagonale è già uguale a 0, si effettuerà un minor numero di prodotti matriciali per la scomposizione.
Nel proseguo del capitolo, sono descritti tre operatori unitari implementati mediante

l'utilizzo di look-up table:

- l'operatore di complemento a due;
- l'operatore di modulo;
- l'operatore di reciproco.

I risultati di scomposizione sono riportati in termini di tempo impiegato e numero di porte quantistiche utilizzato per la descrizione dell'operatore in OpenQASM 2.0. Tutti i suddetti risultati sono stati elaborati solo dopo aver verificato la correttezza funzionale della scomposizione, secondo la metodologia basata sulla simulazione con l'`UnitarySimulator` di Qiskit di IBM [13] e descritta nella Sezione 4.4.

6.1 Operatore di complemento a due

L'operatore di complemento a due (Ca2), su n qubit, può essere rappresentato mediante una generica look-up table con la seguente struttura:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \end{bmatrix} . \tag{6.2}$$

Su tre qubit la matrice diventa:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} . \tag{6.3}$$

Per dimostrare che la matrice è associabile all'operazione di complemento a due, si riportano alcuni esempi di valori di ingresso, analizzando il risultato in uscita dalla matrice.
Si consideri il caso della seconda colonna, rappresentata dall'indice 1[1], che in binario espresso su tre bit, è 001. L'elemento non nullo di questa colonna si trova nell'ultima riga corrispondente alla stringa binaria 111, che in complemento a due rappresenta il numero -1. Si può osservare che 111 è esattamente il complemento a due della stringa binaria 001.
Un altro esempio, si consideri la colonna con indice 5 in binario 101 che in complemento a due rappresenta il numero -3. L'elemento non nullo di quella colonna è in posizione 3 in binario 011; anche in questo caso l'indice di riga dell'elemento non nullo (011) è il complemento a due dell'indice della colonna (101).
È importante evidenziare l'unico caso "anomalo" della matrice rappresentata: il caso della colonna con indice 4. In binario tale indice corrisponde a 100 che in complemento a due è -4; per poter rappresentare correttamente in complemento a due il valore positivo 4 sono necessari quattro qubit (0100). Siccome il numero di qubit in questo caso è tre, non è possibile rappresentare correttamente il risultato. In generale il risultato dell'elemento della colonna in posizione $\frac{N}{2}$ corrispondente alla stringa binaria 10...0 non potrà essere rappresentato correttamente su n qubit, ma necessiterà di un qubit in più.

6.1.1 Implementazione

In questa sotto-sezione è presentato il diagramma di flusso per ottenere la matrice che descrive l'operatore di Ca2.

[1]Per una matrice di dimensione $N \times N$, le righe e le colonne sono rappresentate da indici che vanno da 0 a $N - 1$.

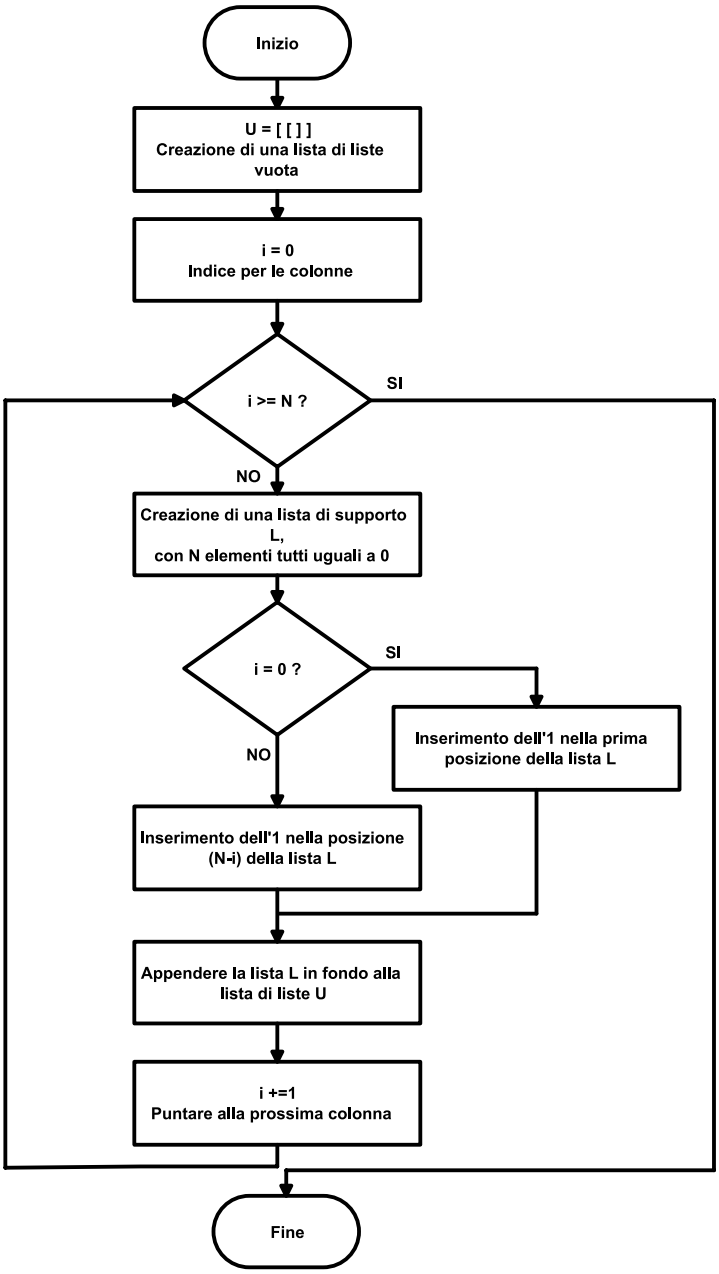


Figura 6.1: Diagramma di flusso per ottenere la matrice che descrive l'operatore di complemento a due.

Nella Figura 6.1 è riportato il diagramma di flusso. La matrice è salvata all'interno della lista di liste U, costruita per righe, utilizzando una lista di supporto L, in cui viene posizionato l'elemento non nullo nella posizione opportuna per ogni colonna.

Alla fine di ogni iterazione la lista L viene appesa in fondo alla lista di liste U. Nella Figura 6.2 è riportato l'esempio di un passaggio per la creazione della lista di liste U.

```
Indice: i = 3
Lista di lista U prima: U = [ [1, 0, 0, 0, 0, 0, 0, 0],
                              [0, 1, 0, 0, 0, 0, 0, 0],
                              [0, 0, 1, 0, 0, 0, 0, 0] ]
Lista di supporto L:    L = [0, 0, 0, 0, 0, 0, 0, 1]
Lista di liste U dopo: U = [ [1, 0, 0, 0, 0, 0, 0, 0],
                             [0, 1, 0, 0, 0, 0, 0, 0],
                             [0, 0, 1, 0, 0, 0, 0, 0],
                             [0, 0, 0, 0, 0, 0, 0, 1] ]
```

Figura 6.2: Esempio della creazione di una linea matrice del complemento a due.

6.1.2 Risultati

Nella seguente sotto-sezione sono riportati i risultati di scomposizione della matrice rappresentante l'operatore di Ca2, confrontando i risultati, in termini di tempo impiegato per la scomposizione e numero di porte utilizzato nella descrizione del circuito in OpenQASM 2.0, con quelli ottenuti per la scomposizione di una matrice casuale piena generica (capitolo 3). È importante sottolineare che per ogni valore di qubit sono stati mediati i risultati di dieci scomposizioni. Inoltre, va precisato che per quanto riguarda i grafici del tempo di scomposizione, viene riportato solamente il tempo totale senza distinguere tra i tempi impiegati nei vari algoritmi di scomposizione; questo perché il tempo impiegato dall'algoritmo di Li-Pelejo risulta essere dominante rispetto alle latenze degli altri due algoritmi.

Nelle Figure 6.3 e 6.4 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del tempo di scomposizione, nel caso di test disabilitati, per una matrice generica piena e per la matrice dell'operatore Ca2. È possibile osservare un aumento della differenza tra le due curve all'aumentare del numero di qubit.

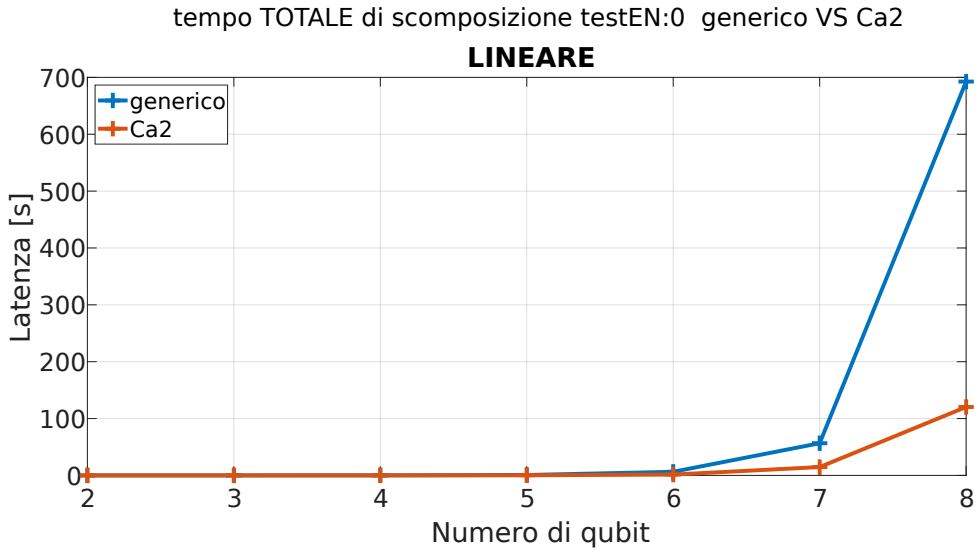


Figura 6.3: Grafico con il tempo totale di scomposizione, per test disabilitati, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala lineare.

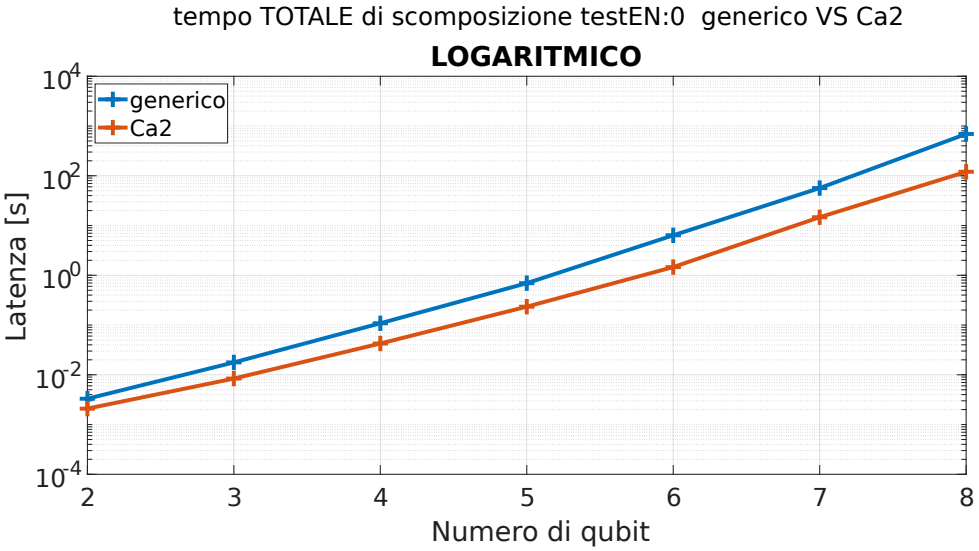


Figura 6.4: Grafico con il tempo totale di scomposizione, per test disabilitati, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala semi-logaritmica.

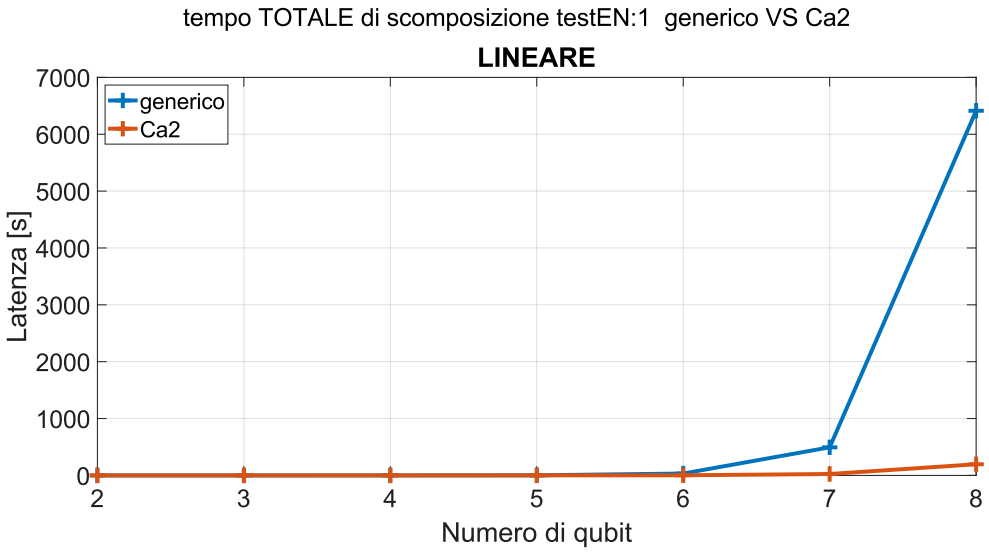


Figura 6.5: Grafico con il tempo totale di scomposizione, per test abilitati, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala lineare.

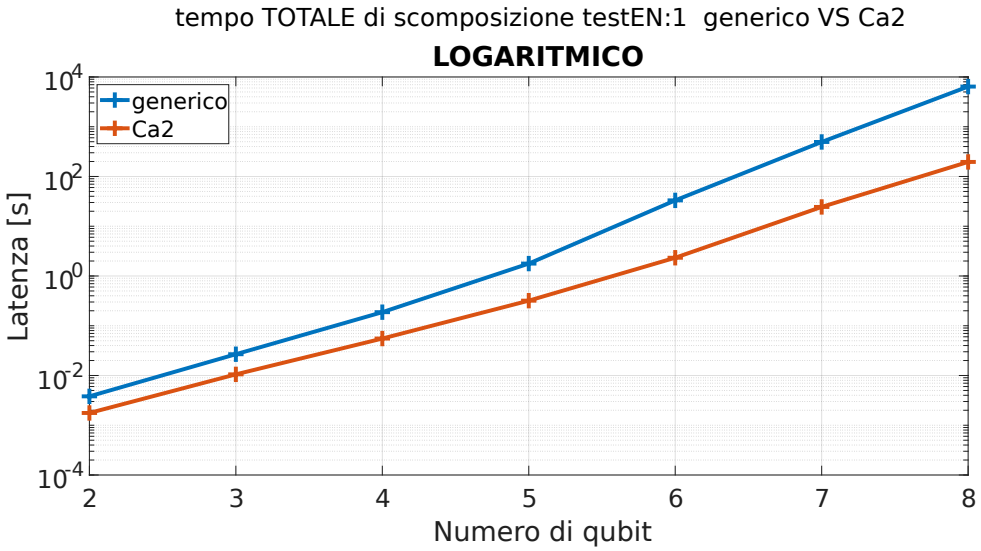


Figura 6.6: Grafico con il tempo totale di scomposizione, per test abilitati, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala semi-logaritmica.

Nelle Figure 6.5 e 6.6 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del tempo di scomposizione, nel caso di test abilitati, per una matrice

generica piena e per la matrice dell'operatore Ca2. È possibile osservare un aumento della differenza tra le due curve all'aumentare del numero di qubit; questa differenza risulta essere ancora più marcata rispetto al caso dei test disabilitati. Infatti nel caso di test abilitati avere un minor numero di matrici implica non solo avere un minor numero di matrici da moltiplicare ma anche un minor numero di matrici da testare, riducendo notevolmente la latenza complessiva.

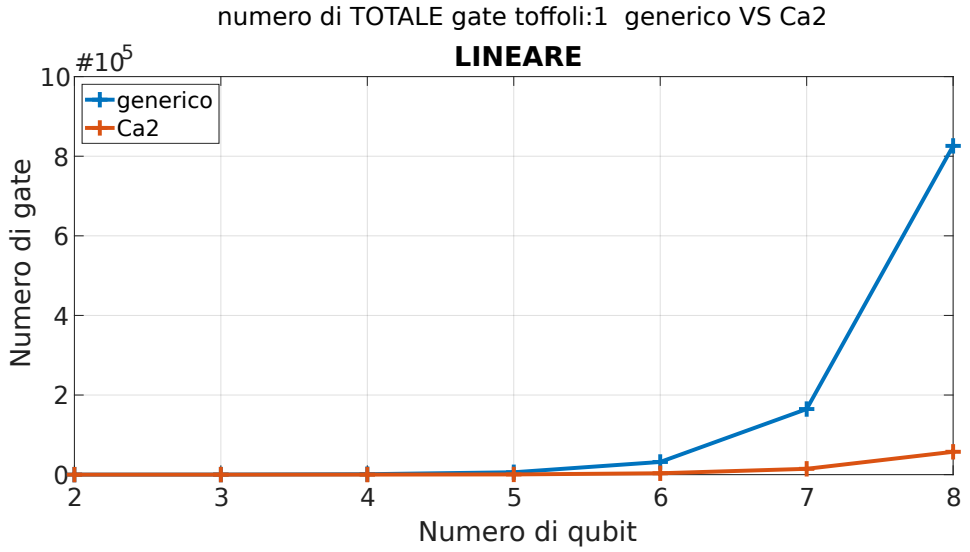


Figura 6.7: Grafico con il numero totale di porte, nel caso di disabilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala lineare.

Nelle Figure 6.7 e 6.8 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del numero di porte quantistiche, nel caso in cui la scomposizione della porte Toffoli sia disabilitata, per una matrice generica piena e per la matrice dell'operatore Ca2. È possibile osservare un aumento della distanza tra le due curve all'aumentare del numero di qubit.

Nelle Figure 6.9 e 6.10 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del numero di porte quantistiche, nel caso in cui la scomposizione della porte Toffoli sia abilitata, per una matrice generica piena e per la matrice dell'operatore Ca2. È possibile osservare una maggiore distanza tra le due curve rispetto al caso in cui la porta Toffoli non sia scomposta. Questo fenomeno è anch'esso

riconducibile al minor numero di matrici generate dalla scomposizione di Li-Pelejo.

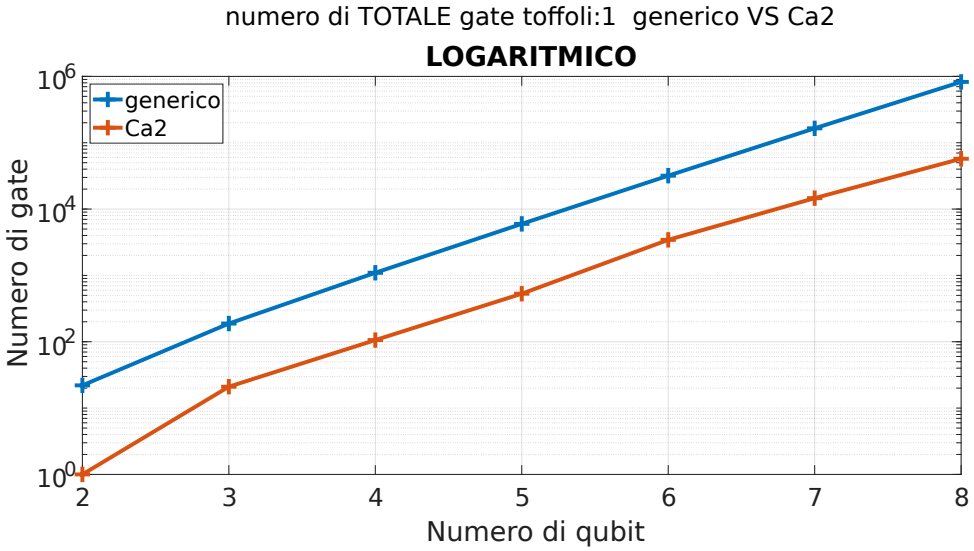


Figura 6.8: Grafico con il numero totale di porte, nel caso di disabilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala semi-logaritmica.

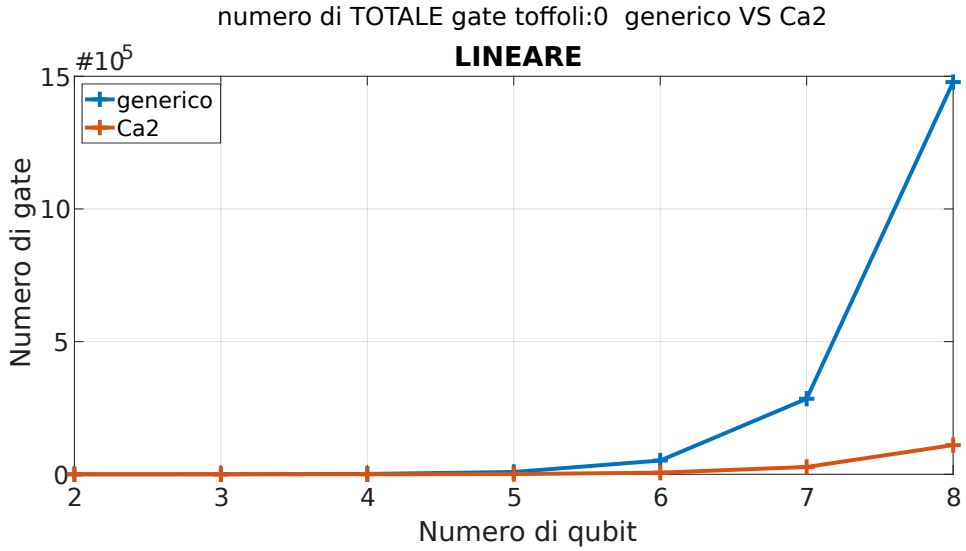


Figura 6.9: Grafico con il numero totale di porte, nel caso di abilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala lineare.

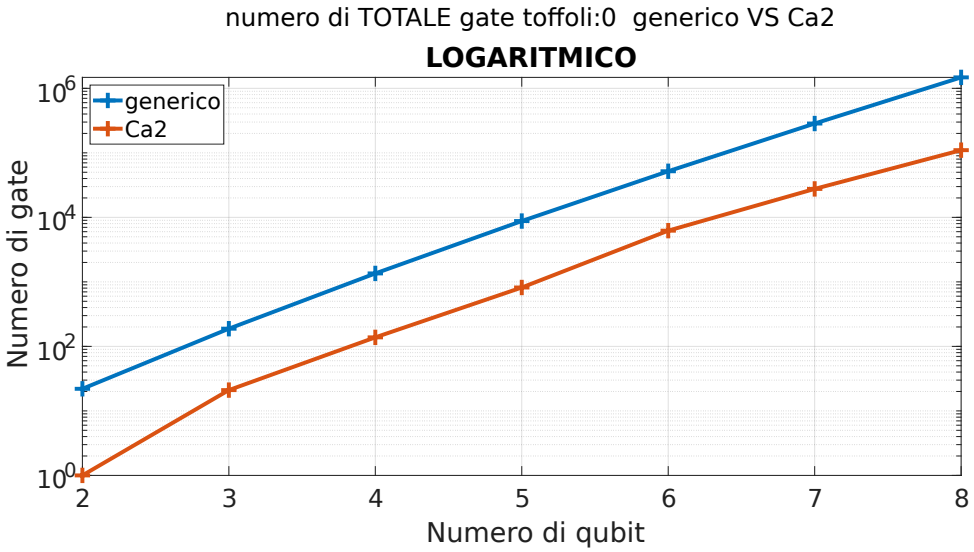


Figura 6.10: Grafico con il numero totale di porte, nel caso di abilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore di Ca2 ed una matrice generica piena, scala semi-logaritmica.

6.2 Operatore di modulo

L'operatore di modulo, su n qubit, può essere rappresentato mediante una generica look-up table con la seguente struttura:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 1 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 & 0 & 0 \end{bmatrix} . \tag{6.4}$$

L'operatore di modulo è a tutti gli effetti l'operatore Ca2 controllato. Infatti, è possibile osservare che la struttura della sua look-up table non è altro che una matrice diagonale a blocchi formata da una matrice identità (evidenziata con il

colore blu in 6.6) e dalla matrice del Ca2 (evidenziata con il colore rosso in 6.6) come sotto-matrici principali e due matrici formate da tutti zeri sull'antidiagonale.

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{Ca2} \end{bmatrix} . \tag{6.5}$$

Su tre qubit la look-up table del modulo è:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} . \tag{6.6}$$

Si riportano alcuni esempi per dimostrare che la matrice riportata sia associabile alla funzione del modulo. È importante sottolineare che la funzione del modulo, rappresentata da da questa look-up table è una funzione controllata della Ca2, per cui non tutti i qubit sono qubit di dato, ma il most significant qubit è un qubit di supporto che controlla l'applicazione dell'operazione di complemento a due. Per l'esattezza il most significant qubit è la copia del secondo qubit più significativo. Questo espediente è necessario per garantire l'unitarietà della matrice, in quanto l'utilizzo tutti i qubit disponibili come qubit di dato non garantisce la rappresentazione della funzione di modulo tramite una matrice unitaria.
Le uniche combinazioni per i qubit, che rappresentano un dato significativo, sono quelle in cui i due qubit più significativi hanno lo stesso valore, ovvero: 000, 001, 110 e 111. Per queste combinazioni l'uscita della look-up table è:

- Per 000 (0 in decimale): l'elemento non nullo si trova nella prima riga corrispondente alla stringa binaria 000 (0 in decimale);
- Per 001 (1 in decimale): l'elemento non nullo si trova nella seconda riga corrispondente alla stringa binaria 001 (1 in decimale);

- Per 110 (-2 in decimale): l'elemento non nullo si trova nella penultima riga corrispondente alla stringa binaria 110 che escludendo il most significant qubit rappresenta il numero 2 in decimale;
- Per 111 (-1 in decimale): l'elemento non nullo si trova nella penultima riga corrispondente alla stringa binaria 101 che escludendo il most significant qubit rappresenta il numero 1 in decimale;

Tutte le altre combinazioni di ingresso non sono significative in quanto il most significant qubit è la copia del secondo bit più significativo, per cui, essi non potranno mai avere un valore diverso tra loro.

6.2.1 Implementazione

In questa sotto-sezione è riportato il diagramma di flusso per la costruzione della look-up table per l'operatore di modulo (Figura 6.11).

Si può osservare che il diagramma di flusso è molto simile a quello proposto per l'operatore Ca2. Questo è dovuto al fatto che la sotto-matrice principale inferiore della look-up table del modulo è proprio la matrice del Ca2.

6.2.2 Risultati

In questa sotto-sezione sono riportati i risultati, in termini di tempo impiegato e numero di porte utilizzate, per la scomposizione della matrice dell'operatore di modulo, confrontandoli con quelli ottenuti per l'operatore di complemento a due. Anche in questo caso i risultati sono stati ottenuti mediando dieci scomposizioni per ogni valore del numero di qubit.

Nelle Figure 6.12 e 6.13 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del tempo di scomposizione, nel caso di test disabilitati, per la matrice dell'operatore Ca2 e per la matrice dell'operatore di modulo. È possibile osservare un leggero aumento della differenza tra le due curve all'aumentare del numero di qubit. Questa somiglianza delle due curve è dovuta al fatto che dal punto di vista della scomposizione metà della matrice del modulo è la matrice della Ca2.

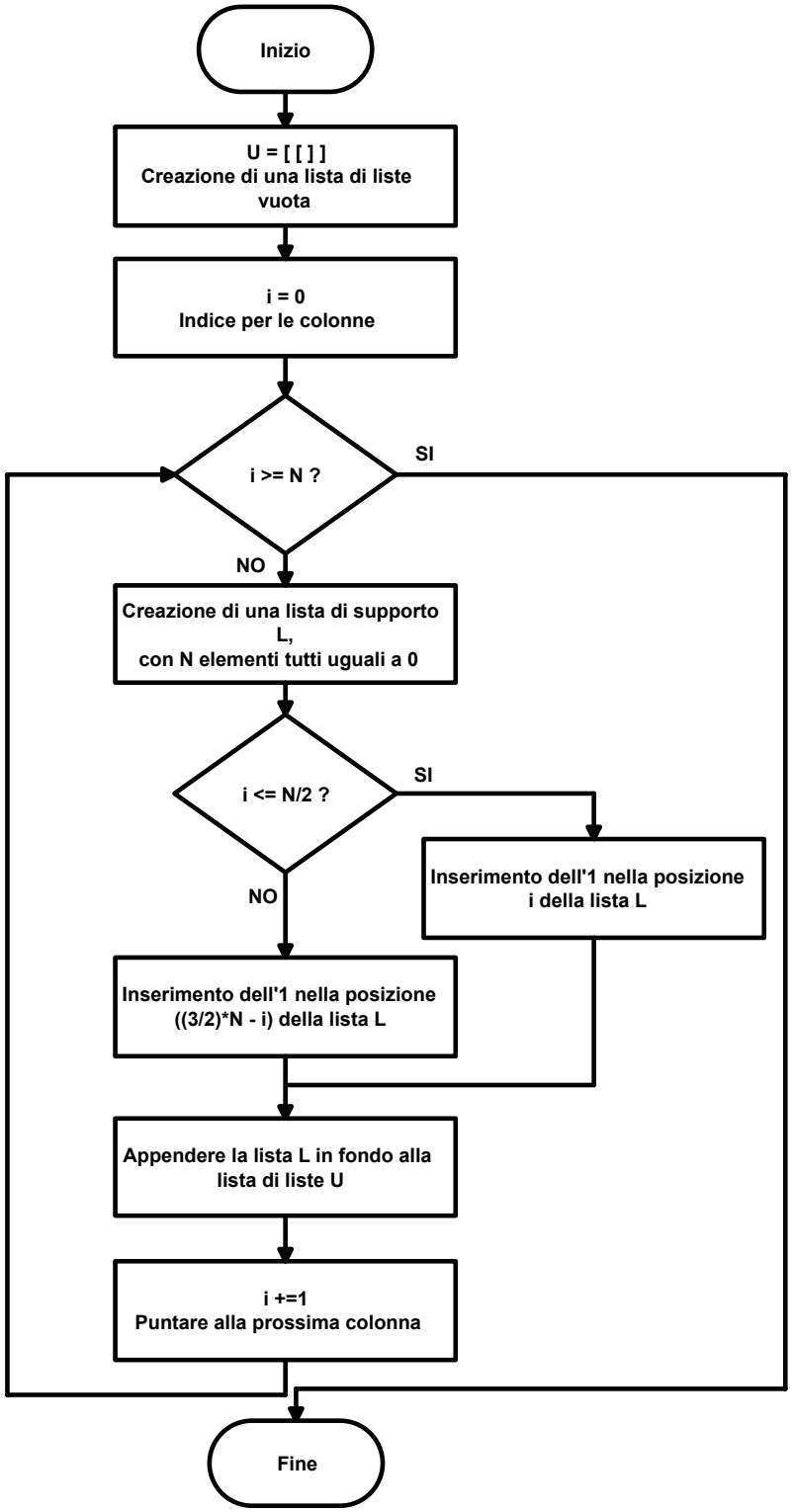


Figura 6.11: Diagramma di flusso per ottenere la matrice che descrive l'operatore di modulo.

È importante osservare che la curva relativa all'operatore di Ca2 scala più velocemente di quella dell'operatore di modulo; questo perché nell'implementazione dell'algoritmo di Li-Pelejo, esso cerca di costruire la matrice di semplificazione anche per gli elementi già nulli, vanificando parte del vantaggio prestazionale ottenibile. Questo perché come già espresso il metodo di scomposizione proposto è una prima versione di una struttura automatica per la generazione di circuiti quantistici partendo da una descrizione tramite matrici unitarie. L'implementazione in questa tesi punta a verificarne la fattibilità e la funzionalità, in futuro questa implementazione potrà essere migliorata ed espansa.

Nelle Figure 6.14 e 6.15 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del tempo di scomposizione, nel caso di test abilitati, per la matrice dell'operatore Ca2 e per la matrice dell'operatore di modulo. È possibile osservare un maggiore aumento della distanza tra le due curve rispetto al caso dei test disabilitati. Per un numero di qubit ridotto (2 e 3) sembra che la scomposizione del modulo richieda più tempo della scomposizione del Ca2 si ritiene che in questi due casi, il tempo di scomposizione misurato sia dipeso fortemente dal carico di lavoro del server su cui sono state fatte le simulazioni.

Nelle Figure 6.16 e 6.17 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del numero di porte quantistiche, nel caso in cui la scomposizione della porte Toffoli sia disabilitata, per la matrice dell'operatore modulo e per quella del Ca2. Si può osservare che nel caso di due qubit, la matrice del modulo è la matrice identità, per cui il numero di porte utilizzato per la scomposizione è 0.

Nelle Figure 6.18 e 6.19 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del numero di porte quantistiche, nel caso in cui la scomposizione della porte Toffoli sia abilitata, per la matrice dell'operatore modulo e per quella del Ca2. Anche in questo caso, come nel caso in cui la scomposizione della porta Toffoli si a disabilitato, la distanza tra le due curve cresce all'aumentare del numero di qubit.

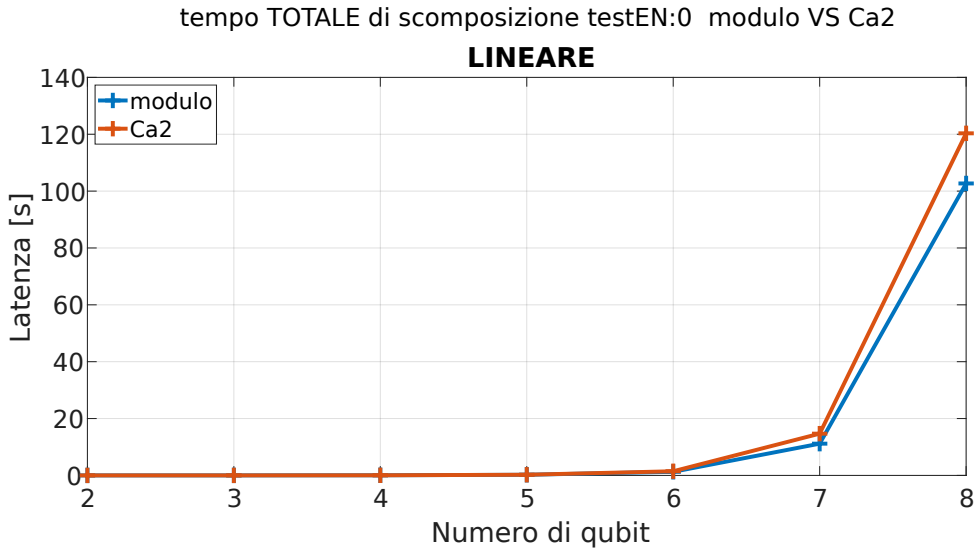


Figura 6.12: Grafico con il tempo totale di scomposizione, per test disabilitati, della matrice rappresentate l'operatore di modulo e la matrice dell'operatore di Ca2, scala lineare.

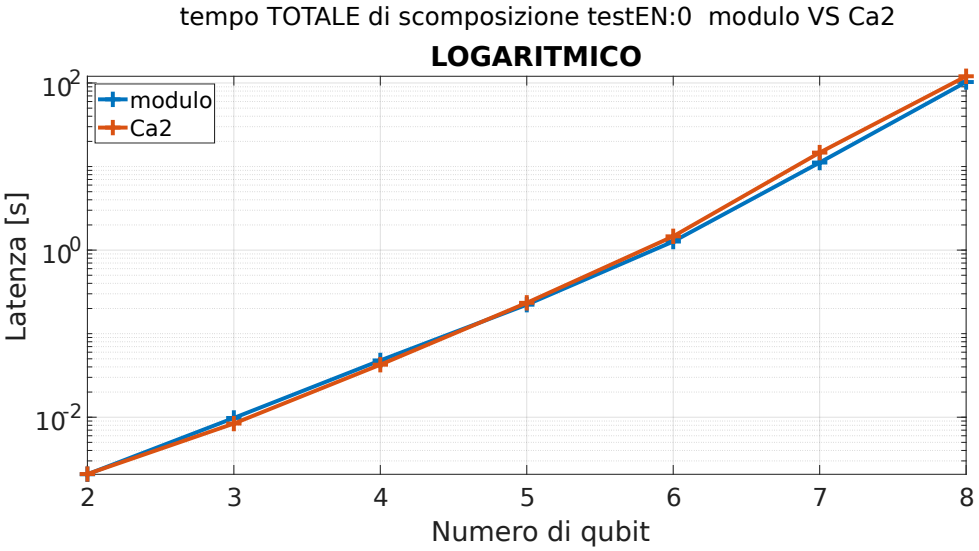


Figura 6.13: Grafico con il tempo totale di scomposizione, per test disabilitati, della matrice rappresentate l'operatore di modulo e la matrice dell'operatore di Ca2, scala semi-logaritmica.

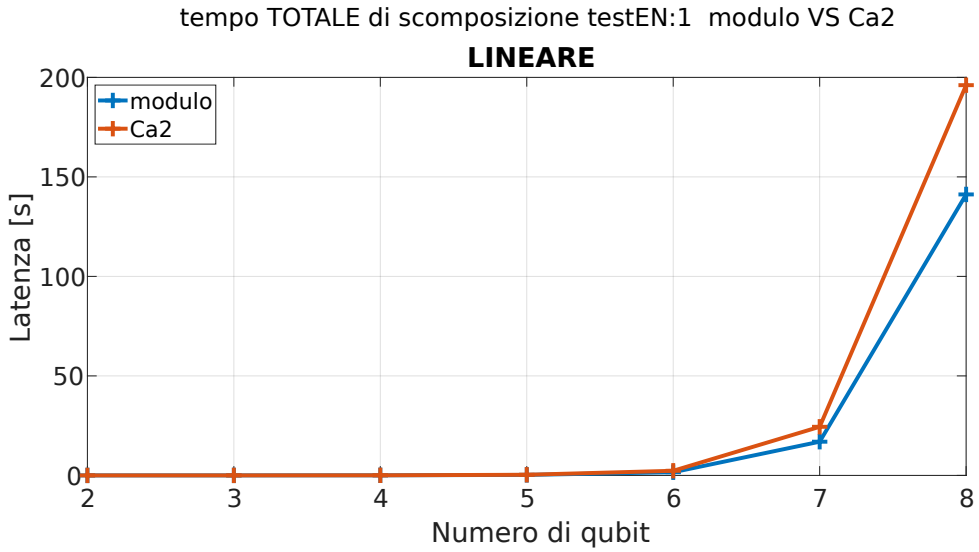


Figura 6.14: Grafico con il tempo totale di scomposizione, per test abilitati, della matrice rappresentate l'operatore di modulo e la matrice dell'operatore di Ca2, scala lineare.

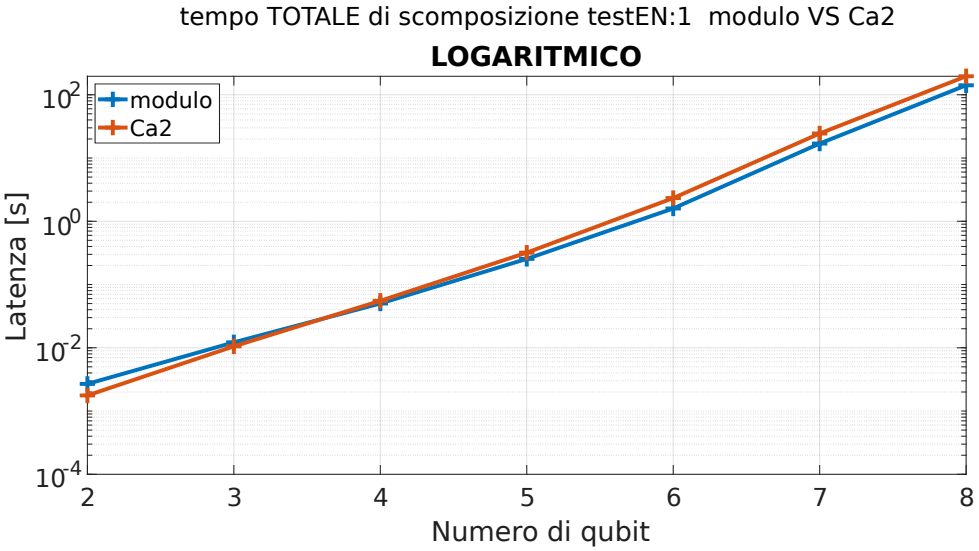


Figura 6.15: Grafico con il tempo totale di scomposizione, per test abilitati, della matrice rappresentate l'operatore di modulo e la matrice dell'operatore di Ca2, scala semi-logaritmica.

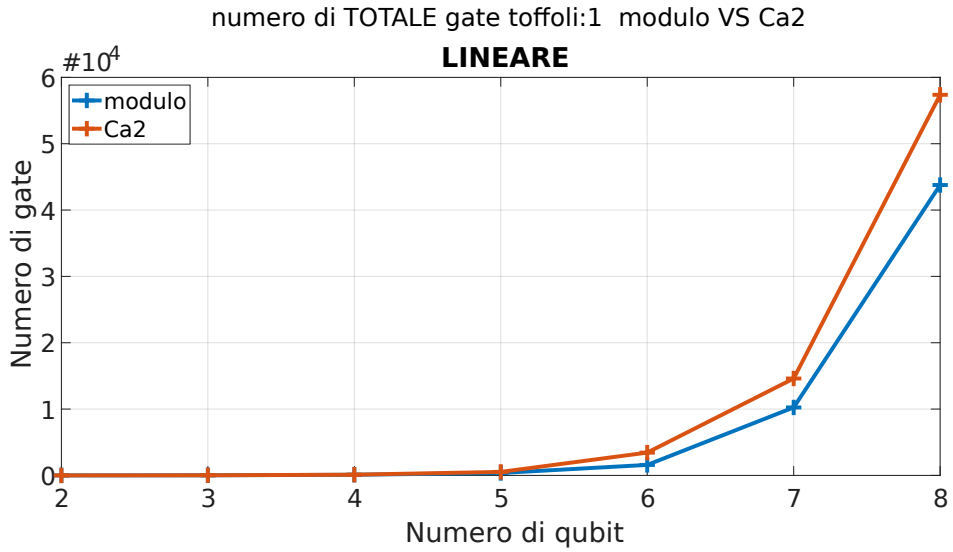


Figura 6.16: Grafico con il numero totale di porte, nel caso di disabilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore del modulo e quella della Ca2, scala lineare.

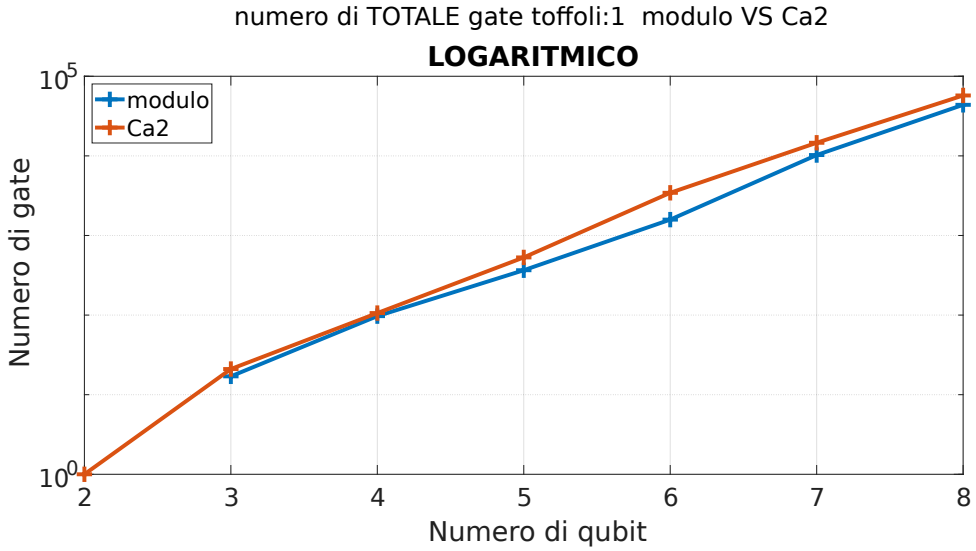


Figura 6.17: Grafico con il numero totale di porte, nel caso di disabilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore del modulo e quella della Ca2, scala semi-logaritmica.

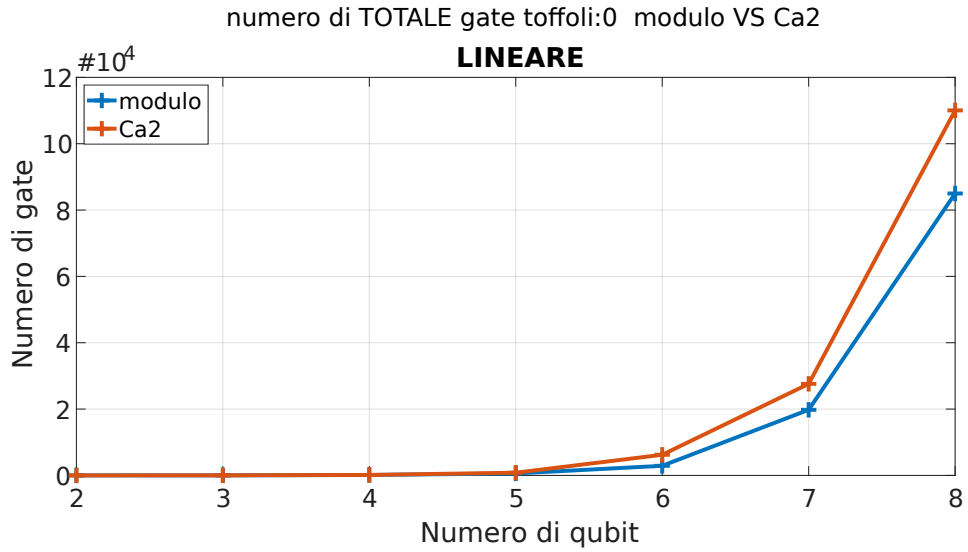


Figura 6.18: Grafico con il numero totale di porte, nel caso di abilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore del modulo e quella della Ca2, scala lineare.

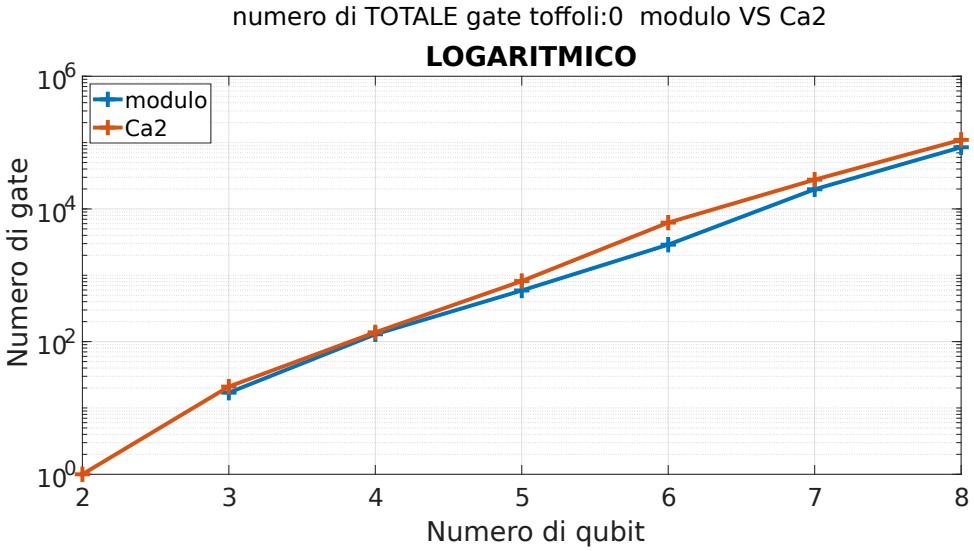


Figura 6.19: Grafico con il numero totale di porte, nel caso di abilitazione della scomposizione della porta Toffoli, della matrice rappresentate l'operatore del modulo e quella della Ca2, scala semi-logaritmica.

6.3 Operatore di reciproco

L'operatore di reciproco, su tre qubit può essere rappresentato mediante la seguente look-up table:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} . \tag{6.7}$$

Gli indici delle colonne sono da considerarsi come valori interi in complemento a due, mentre gli indici delle righe devono essere considerati come valori in complemento a due in virgola fissa, dove il most significant qubit rappresenta la parte intera e tutti gli altri qubit rappresentano la parte frazionaria.

C'è da considerare che dato un numero generico n (maggiore di 2) di qubit, non è possibile individuare una matrice unitaria delle dimensione $2^n \times 2^n$ che rappresenti accuratamente la funzione del reciproco. Questo perché è necessario disporre di un maggiore numero di righe, nella matrice, per essere in grado di inserire un solo 1 per colonna. Da un punto di vista numerico, il problema risiede nel fatto che la risoluzione ottenibile da una matrice $2^n \times 2^n$ non è sufficiente per rappresentare univocamente i reciproci per tutti i valori possibili nel range da -2^{n-1} a $2^{n-1} - 1$. Per chiarire meglio il problema viene riportato l'esempio nel caso di n uguale a tre qubit.

La matrice su tre qubit è riportata sopra 6.7. Con la convenzione adottata, contando nell'ordine da destra verso sinistra, gli indici delle colonne rappresentano i valori in complemento a due: 0,1,2,3,-4,-3,-2,-1. Mentre gli indici delle righe, in ordine dall'alto verso il basso, sono i valori: 0, $\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$, -1, $-\frac{3}{4}$, $-\frac{1}{2}$ e $-\frac{1}{4}$. Si può notare che per quanto riguarda la colonna con valore -3 (la sesta colonna contando da destra a sinistra) l'indice relativo alla all'elemento non nullo rappresenta il valore $-\frac{1}{2}$, che è un approssimazione molto grossolana del valore desiderato $-\frac{1}{3}$. Invece,

per la colonna con valore -2 (la penultima colonna contando da destra a sinistra), l'indice dell'elemento non nullo, rappresenta il valore $-\frac{3}{4}$, un approssimazione molto grossolana del valore esatto $-\frac{1}{2}$. Quello che è accaduto, è che il valore $-\frac{1}{2}$ è stato "preso" dalla colonna -3; questo perché non vi è "abbastanza spazio" tra la riga con valore $-\frac{1}{4}$ e quella con valore $-\frac{1}{2}$ per inserire un'approssimazione del valore $-\frac{1}{3}$. Per superare questo problema è necessario inserire dei qubit di supporto come meno significativi per aumentare la risoluzione.
Il numero di qubit di supporto può essere calcolato considerando la risoluzione massima necessaria per rappresentare correttamente tutti i numeri nell'intervallo da -2^{n-1} a $2^{n-1}-1$. Tale risoluzione può essere calcolata come:

$$\max_ris = \frac{1}{2^{n-1}-1} - \frac{1}{2^{n-1}} = \frac{1}{2^{2(n-1)}-2^{n-1}}\,. \tag{6.8}$$

Per $n >> 1$ la formula può essere semplificata come:

$$\max_ris \approx \frac{1}{2^{2(n-1)}}\,. \tag{6.9}$$

Da cui il numero totale di qubit necessari per rappresentare correttamente tutti i valori è:

$$n_qubit_tot = 2\cdot(n-1)+1 = 2n-1\,. \tag{6.10}$$

Per cui il numero di qubit di supporto è dato da:

$$n_supp = n_qubit_tot - n = 2n-1-n = n-1\,. \tag{6.11}$$

Per esempio, nel caso di tre qubit saranno necessari due qubit di supporto.
La necessità dei qubit di supporto fa crescere notevolmente le dimensioni delle matrici adoperate; infatti esse avranno dimensioni $2^{n+n_supp}\times 2^{n+n_supp}$. Questo può risultare svantaggioso in termini di prestazioni dell'algoritmo di scomposizione. Tuttavia, la grande quantità di zeri presenti nella matrice compensa parzialmente questo peggioramento.
La matrice riportata ad inizio sezione (6.7) è la matrice del reciproco con due qubit di dato ed uno di supporto.

6.3.1 Implementazione

In questa sotto-sezione è riportato il diagramma di flusso per generare la look-up table per l'operatore del reciproco (Figura 6.20).

L'algoritmo di costruzione della matrice del reciproco, parte dalla matrice identità e "sposta" gli 1 delle colonne con indici "significativi", sulle righe opportune.
Considerando che il valore di tutti i qubit di supporto è 0, gli indici significativi sono tutti quei valori di indice che presentano tutti i valori dei qubit di supporto uguale a 0; per esempio nel caso di tre qubit, con due qubit di dato e uno di supporto i valori degli indici validi sono: 000, 010, 100 e 110, dove il qubit di supporto è il meno significativo. Per spostarsi da un indice significativo al successivo basta incrementare il puntatore delle colonne di 2^{n_supp} dove n_supp è il numero di qubit di supporto.
L'algoritmo itera sulla prima metà (la parte di matrice che riguardante i numeri positivi) delle colonne e modifica, di conseguenza, anche la seconda metà (la parte di matrice che riguardante i numeri negativi) sfruttando la simmetria della look-up table dell'operatore del reciproco.
L'indice di riga in cui inserire l'elemento non nullo per ogni colonna viene calcolato tramite la seguente formula:

$$riga = \left\lceil 2^{n_qubit_tot-1} \cdot \frac{2^{n_supp}}{colonna} \right\rceil . \tag{6.12}$$

Per spiegare il significato della formula si utilizza il seguente esempio: nel caso di cinque qubit, con tre qubit di dato e due di supporto, la colonna corrispondente al valore decimale 3 è la colonna con indice 12 (questo può essere calcolato come $3 \cdot 2^{n_supp} = 3 \cdot 2^2 = 12$); per cui la frazione $\frac{2^{n_supp}}{colonna}$ risulta essere $\frac{2^2}{12} = \frac{1}{3}$, che è il valore di reciproco desiderato. Tale valore viene scalato sul numero di qubit totali a disposizione ($2^{n_qubit_tot-1} \cdot \frac{1}{3} = 2^{5-1} \cdot \frac{1}{3} \cong 5.334$), e ne viene preso il valore intero immediatamente successivo ($\lceil 5.334 \rceil = 6$). Eseguendo il calcolo si ottiene l'indice di riga uguale a 6 che corrisponde al valore $\frac{3}{8}$ (0.375) che è la corretta l'approssimazione di $\frac{1}{3}$ (0.333).

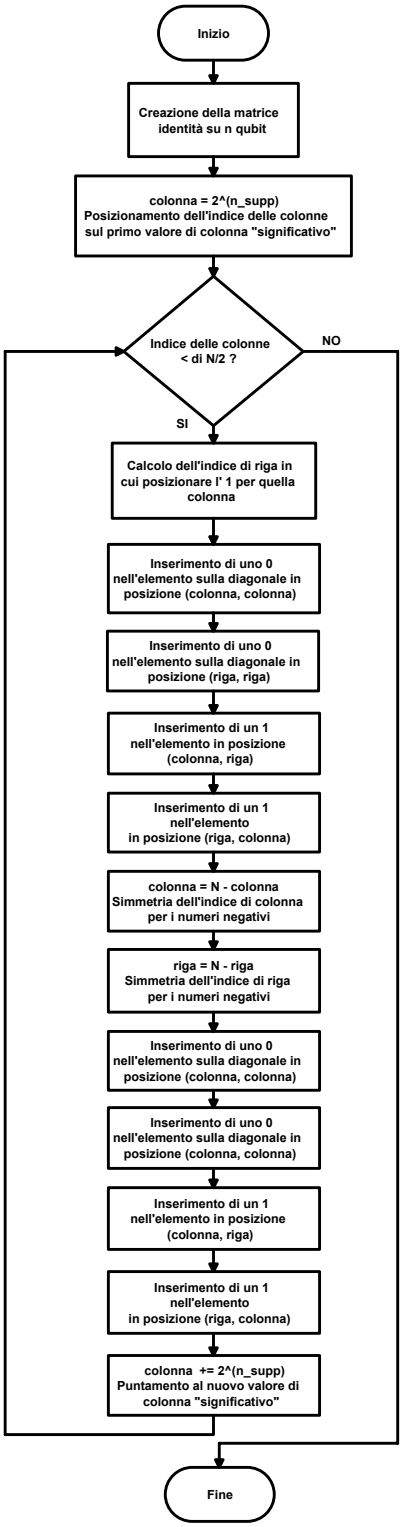


Figura 6.20: Diagramma di flusso per ottenere la matrice che descrive l'operatore del reciproco.

6.3.2 Risultati

In questa sotto-sezione sono riportati i risultati della scomposizione della matrice del reciproco, in termini di tempo impiegato e numero di porte quantistiche utilizzato per la descrizione. Anche in questo caso i risultati riportati nei grafici sono stati ottenuti mediando dieci scomposizioni per ogni valore del numero di qubit.

Nelle Figure 6.21 e 6.22 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del tempo di scomposizione, nel caso di test disabilitati, per la matrice dell'operatore reciproco. Nei grafici sono confrontati tutti i casi possibili di scomposizione. Si può osservare che i casi peggiori sono quelli con test abilitati. Inoltre, è possibile constatare che non vi è grande differenza di tempo tra l'abilitazione e la disabilitazione della scomposizione della porta Toffoli; a dimostrazione del fatto che il tempo impiegato dalla fase di scomposizione dell'algoritmo di Li-Pelejo è dominante rispetto a quello occupato dalle altre due fasi di scomposizione.

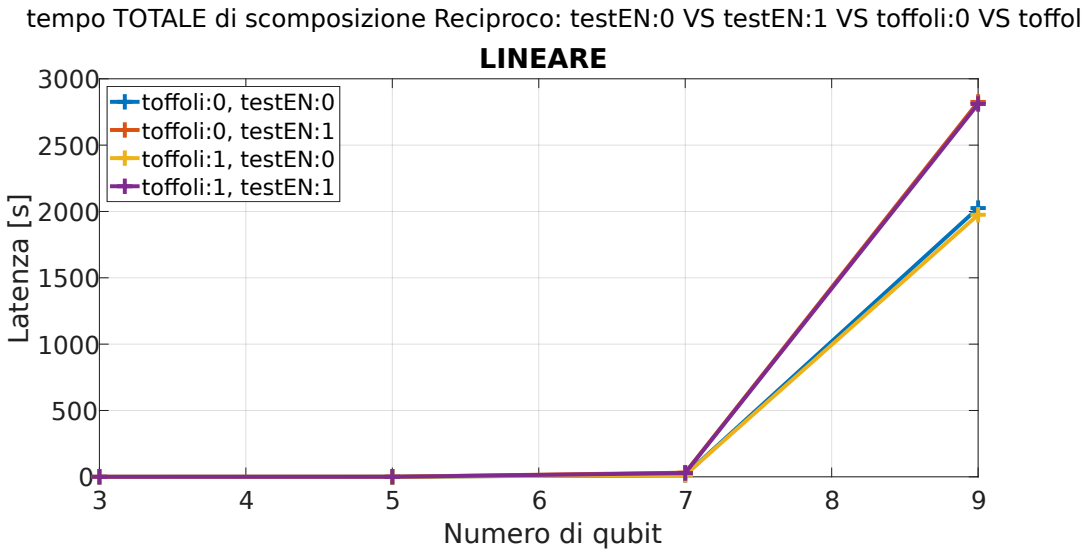


Figura 6.21: Grafico con il tempo di scomposizione dell'operatore reciproco, in scala lineare.

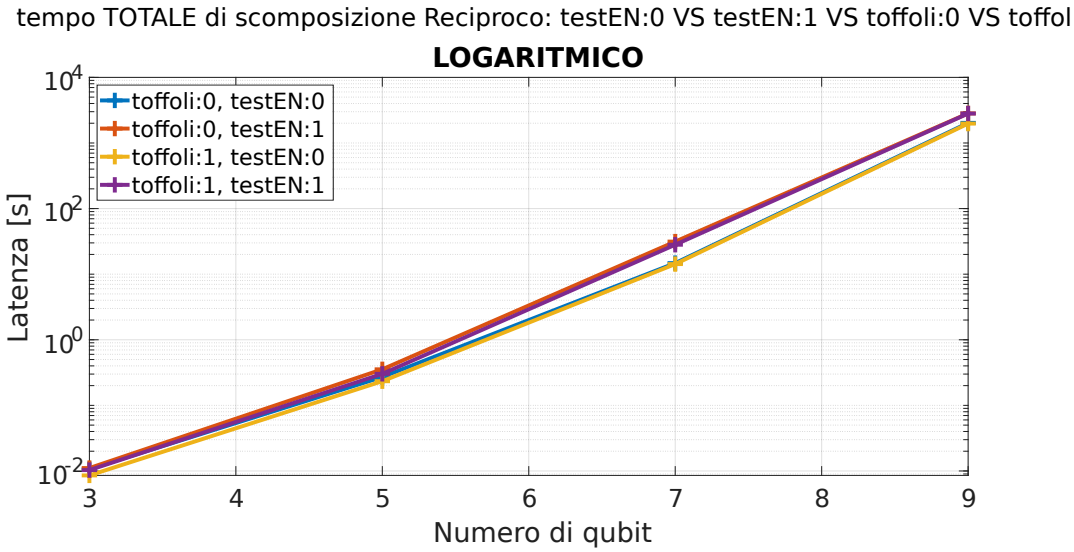


Figura 6.22: Grafico con il tempo di scomposizione dell'operatore reciproco, in scala semi-logaritmica.

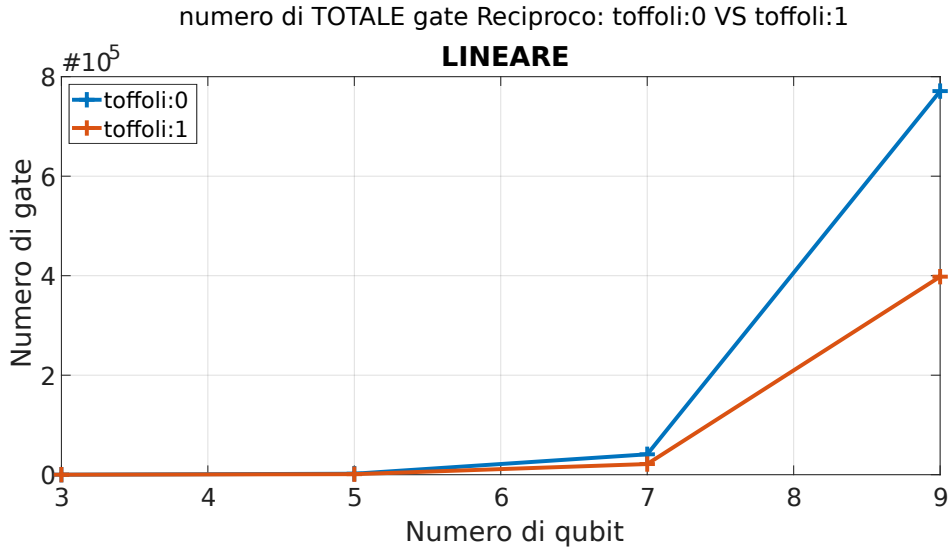


Figura 6.23: Grafico con il numero totale di porte utilizzate nella scomposizione dell'operatore reciproco, in scala lineare.

Nelle Figure 6.23 e 6.24 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) del numero di porte quantistiche, nel caso in cui la scomposizione

della porte Toffoli sia abilitata e disabilitata, per la matrice dell'operatore reciproco. Anche in questo caso si può notare un aumento della distanza tra le due curve all'aumentare del numero di qubit.

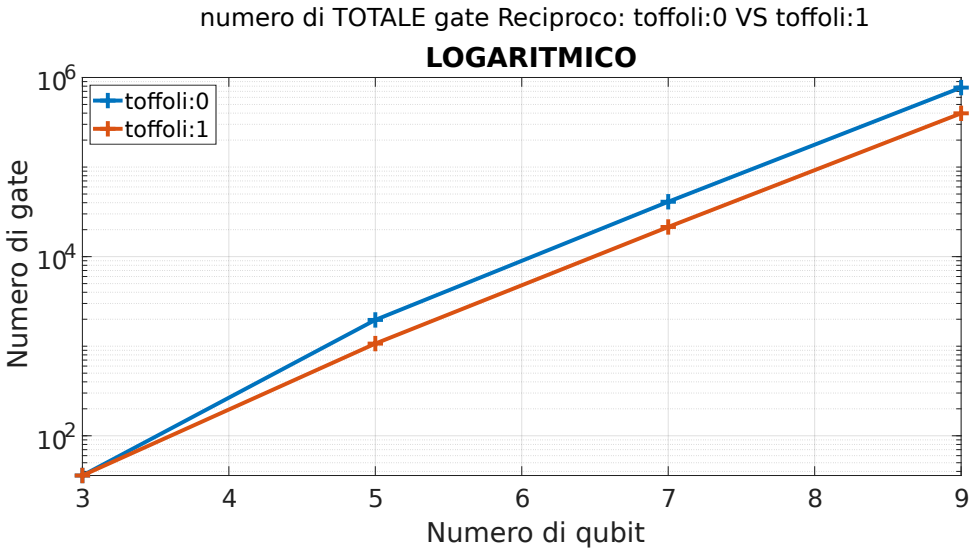


Figura 6.24: Grafico con il numero totale di porte utilizzate nella scomposizione dell'operatore reciproco, in scala semi-logaritmica.

Confronto del tempo totale di scomposizione tra i tre operatori analizzati Nelle Figure 6.25 e 6.26 sono riportati i grafici (rispettivamente in scala lineare e semi-logaritmica) dell'interpolazione dei dati del tempo di scomposizione dei tre operatori analizzati, nel caso di test abilitati e scomposizione della porta Toffoli abilitata.

Dai grafici si può osservare che circa oltre 6 qubit, il tempo utilizzato per la scomposizione dell'operatore di reciproco, supera quello degli altri due operatori. Questo è dovuto al fatto che, nonostante nella struttura della look-up table del Ca2 (e conseguentemente anche in quella del modulo) il numero di elementi non nulli fuori dalla diagonale sia maggiore rispetto a quello dell'operatore del reciproco nella matrice del Ca2 c'è un certo "ordine" che permette all'algoritmo di Li-Pelejo di utilizzare un unica matrice per eliminare due elementi fuori dalla diagonale. Questa proprietà della matrice si palesa maggiormente all'aumentare del numero di qubit.

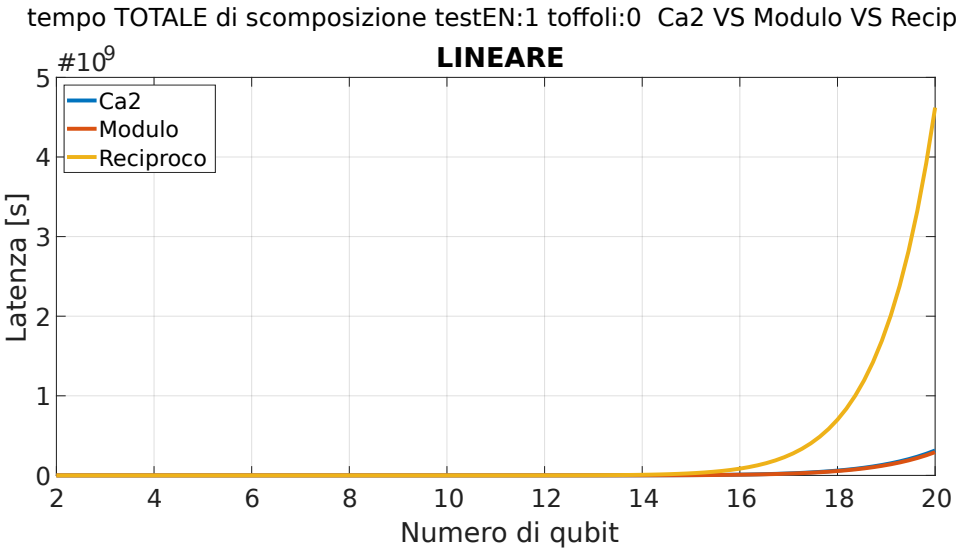


Figura 6.25: Grafico con l'interpolazione del tempo totale di scomposizione degli operatori descritti da look-up table analizzati, in scala lineare.

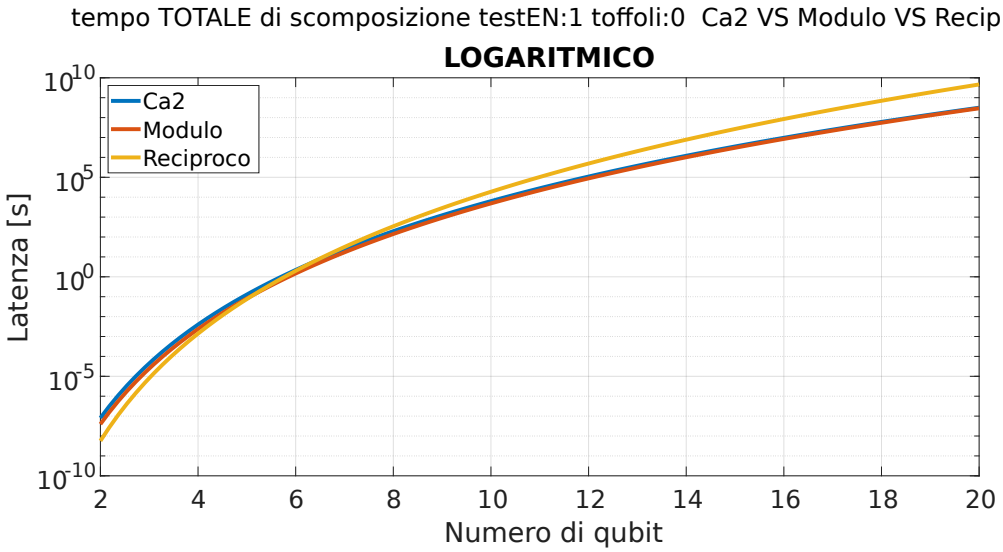


Figura 6.26: Grafico con l'interpolazione del tempo totale di scomposizione degli operatori descritti da look-up table analizzati, in scala semi-logaritmica.

Capitolo 7

Conclusioni e prospettive future

In questo capitolo è presentata un'analisi conclusiva dell'algoritmo di scomposizione delle matrici unitarie proposto ed implementato, aggiungendo considerazioni sulle possibili migliorie dell'algoritmo che potrebbero essere introdotte in futuro.

Il metodo proposto adopera tre passi principali per scomporre una matrice unitaria in un circuito quantistico, descritto tramite il linguaggio OpenQASM 2.0. I tre passi di scomposizione sono:

- L'algoritmo di Li-Pelejo: usato per scomporre una matrice unitaria in un circuito formato da sole porte non controllate o porte di tipo multi-controllo singolo-target;
- L'algoritmo ricorsivo di Nielsen-Chuang: usato per scomporre le porte multi-controllo singolo-target in una serie di porte singolo-controllo singolo-target;
- Il recupero di fase: usato per descrivere la rotazione delle porte controllate di tipo $U1$ e $U3$, in maniera compatibile con il linguaggio dell'OpenQASM 2.0.

È stato constatato che aumentando il numero di qubit su cui lavora la matrice unitaria, aumenta esponenzialmente il tempo necessario per la scomposizione. In particolare, aumenta la latenza dell'algoritmo di Li-Pelejo, che risulta essere dominante su quelle degli altri due algoritmi. Il tempo impiegato dall'algoritmo di Li-Pelejo aumenta in quanto aumentano esponenzialmente le dimensioni delle matrici su cui questo algoritmo lavora ed aumenta anche il numero di matrici (e contestualmente

dei prodotti matriciali) necessarie per la scomposizione. Inoltre, l'abilitazione dei test, per verificare che l'algoritmo generi matrici unitarie con un errore inferiore ad una certa soglia impostabile dall'utente, aumenta ulteriormente il tempo totale.
Per ridurre la latenza della scomposizione, si potrebbe agire sia a livello software sia a livello hardware. Nel primo caso, si potrebbe puntare a parallelizzare l'esecuzione dell'algoritmo, sopratutto la parte relativa ai test sulle matrici, che aumentano considerevolmente il tempo di scomposizione. Infatti, ogni elemento della matrice può essere testato indipendentemente dagli altri, favorendo così la possibilità di utilizzo di tecniche avanzate di *multi-threading* per ridurre la latenza totale [14]. Un'altra opzione valida è quella di optare per una rappresentazione numerica in virgola fissa al posto di quella attuale *floating-point*, al fine di velocizzare le operazioni aritmetiche ed algebriche. Tuttavia, bisogna ricordare che tale scelta potrebbe comportare un aumento dell'errore numerico.
Per quanto concerne l'hardware, si potrebbe eseguire la scomposizione su un processori di tipo Single-Instruction Multiple-Data (SIMD), come le Graphics Processing Unit (GPU), che risultano affidabili ed efficienti con dati in forma matriciale [15].

Oltre al tempo necessario per la scomposizione è stato analizzato il numero di porte quantistiche utilizzate per la scrittura del circuito in OpenQASM 2.0. Anche in questo caso, il numero di porte utilizzato aumenta esponenzialmente con il numero di qubit su cui lavora la matrice di ingresso all'algoritmo. Questo aumento del numero di porte è principalmente dovuto alla scomposizione di Nielsen-Chuang ed al recupero di fase. Infatti, il primo aggiunge cinque porte per ogni controllo eliminato da una porta multi-controllo singolo-target; inoltre se la porta in questione ha più di due controlli, tre delle porte aggiunte dall'algoritmo saranno porte di tipo multi-controllo singolo-target e dovranno essere scomposte ricorsivamente dall'algoritmo. Per quanto riguarda il recupero di fase, esso sostituisce ogni porta singolo-controllo singolo-target con cinque porte di tipo X, C$U1$ e C$U3$. Questo aumento del numero di porte è ancora più marcato nel caso in cui si decida di scomporre anche la porta di Toffoli.
Per ridurre il numero di porte si potrebbe fornire l'OpenQASM 2.0 generato in ingresso ad un compilatore quantistico, in grado di compattare il circuito, identificando dei pattern di porte equivalenti ad altri circuiti, ma richiedenti un numero di risorse inferiore.

In ogni caso, è importante precisare che l'infrastruttura software è stata sviluppata per garantire l'integrazione futura di algoritmi di scomposizione delle matrici unitarie diversi da quello descritto. Il confronto dei risultati forniti da questi algoritmi potrebbe permettere di stabilire dei casi d'uso in cui ciascuno di questi si comporta meglio degli altri, in termini di latenza e/o di numero di porte. Per esempio, durante lo svolgimento della tesi è stato constatato che l'infrastruttura proposta scompone matrici sparse, come ad esempio quelle di tipo *look-up table* usate per descrivere operatori aritmetici e logici, in tempo minore di quello con matrici dense, poiché sono presenti molti elementi nulli al loro interno. Si ritiene, dunque, che il metodo attuale sia generalmente più adatto alla scomposizione di queste matrici. Alla luce di appena scritto, la libreria del VLSI Lab potrebbe disporre in futuro di più generatori di circuiti quantistici in OpenQASM 2.0 tra cui scegliere, in funzione della matrice da scomporre, quindi dell'applicazione.
Sebbene vi siano delle funzionalità da migliorare, ci si augura che l'attività descritta in questa tesi possa essere considerata un ragionevole punto di partenza per lo sviluppo di una libreria portabile e modulare di circuiti quantistici descritti in modo comportamentale, con la consapevolezza che essa è già in grado di fornire una scomposizione funzionalmente corretta. L'auspicio è che l'intera infrastruttura sia utilizzabile per progettare circuiti ed applicazioni finalizzati alla risoluzione di problemi applicativi del mondo reale.

Bibliografia

[1] N. S. Yanofsky e M. A. Mannucci, *Quantum computing for computer scientists*. Cambridge University Press, 2008. DOI: 10.1017/CBO9780511813887.

[2] M. A. Nielsen e I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[3] L. M. K. Vandersypen e I. L. Chuang, «NMR techniques for quantum control and computation,» *Rev. Mod. Phys.*, vol. 76, pp. 1037–1069, 4 gen. 2005. DOI: 10.1103/RevModPhys.76.1037.

[4] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson e W. D. Oliver, «A quantum engineer's guide to superconducting qubits,» *Applied Physics Reviews*, vol. 6, n. 2, p. 021 318, giu. 2019. DOI: 10.1063/1.5089550.

[5] L. K. Grover, «Quantum Mechanics Helps in Searching for a Needle in a Haystack,» *Phys. Rev. Lett.*, vol. 79, pp. 325–328, 2 lug. 1997. DOI: 10.1103/PhysRevLett.79.325.

[6] P. W. Shor, «Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer,» *SIAM Review*, vol. 41, n. 2, pp. 303–332, gen. 1999. DOI: 10.1137/s0036144598347011.

[7] A. W. Cross, L. S. Bishop, J. A. Smolin e J. M. Gambetta, «Open quantum assembly language,» *arXiv preprint*, 2017. indirizzo: https://arxiv.org/abs/1707.03429v2.

[8] L. Raggi, *Arithmetic circuits for quantum computing: a software library*, Tesi di laurea magistrale in Ingegneria Elettronica, 2020. indirizzo: https://webthesis.biblio.polito.it/15853/.

[9] C. K. Li e D. C. Pelejo, «Decomposition of quantum gates,» *International Journal of Quantum Information*, vol. 12, n. 01, p. 1 450 002, 2014. DOI: 10.1142/S0219749914500026.

[10] S. Andersson, A. Asfaw, A. Corcoles, L. Bello, Y. Ben-Haim et al. «Learn Quantum Computation Using Qiskit.» (2020), indirizzo: https://qiskit.org/textbook/ch-gates/phase-kickback.html.

[11] A. S. Householder, «Unitary Triangularization of a Nonsymmetric Matrix,» *J. ACM*, vol. 5, n. 4, pp. 339–342, ott. 1958, ISSN: 0004-5411. DOI: 10.1145/320941.320947.

[12] H. Anton e C. Rorres, *Elementary linear algebra: applications version*. John Wiley & Sons, 2013.

[13] H. Abraham et al., *Qiskit: An Open-source Framework for Quantum Computing*, 2021. DOI: 10.5281/zenodo.2573505.

[14] B. Marker, F. G. Van Zee, K. Goto, G. Quintana-Orti e R. A. Van De Geijn, «Toward scalable matrix multiply on multithreaded architectures,» in *European Conference on Parallel Processing*, Springer, 2007, pp. 748–757. DOI: 10.1007/978-3-540-74466-5_79.

[15] K. Fatahalian, J. Sugerman e P. Hanrahan, «Understanding the efficiency of GPU algorithms for matrix-matrix multiplication,» in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, 2004, pp. 133–137. DOI: 10.1145/1058129.1058148.