SQCS 學生量子電腦交流會量子專題競賽

作品說明書

QPE 電路優化與 MIMO 系統特徵值分析應用：跨越 Qiskit 模擬與 FPGA - PSK 信號量測實作

組名： QPEX Lab

成員：鐘婉庭

指導教授：沈瑞欽

目錄

壹、 簡介

在無線通訊中，訊號傳播常因建築物遮蔽、多重反射或移動環境產生路徑差異，導致接收訊號強度不穩定，形成所謂「衰落通道」，其中以 Rayleigh 衰落最具代表性。此現象對振幅調變特別不利，易造成訊號失真與解調錯誤[1]。

相較之下，相位偏移調變（Phase Shift Keying, PSK）因具抗雜訊能力與頻譜效率高的優勢[2]，較能應對此類通道，故廣泛應用於低功耗與長距離通訊場景。然而，傳統 PSK 接收器需仰賴混頻與載波同步機制，當訊號功率降低時，易導致相位同步失準，錯誤率大幅上升[1]，限制其於極端環境下的應用潛能。

在此背景下，量子相位估計（Quantum Phase Estimation, QPE）被視為一項具突破性的演算法，其不依賴信號強度，而是透過量子疊加、干涉與逆量子傅立葉轉換，可將特徵值中隱含的相位資訊轉換為可測量的位元結果。QPE 已被廣泛應用於 Shor's 演算法、Hamiltonian 模擬等量子計算領域，作為估算量子系統特徵值的核心技術[3]，展現出高精度與高度通用性。

貳、 研究動機與目的

儘管 QPE 在理論上具高度潛力，但在實作上面臨幾個重大挑戰，尤其是受控單位元閘門的重複堆疊與電路深度呈指數成長，導致資源需求龐大、效率受限，難以應用於較大規模的量子系統。

為克服此瓶頸，Lu 等人於 2025 年提出 Lu's Optimized Generator for QPE 架構，透過 Peter–Weyl 分解與多執行緒平行化設計，優化受控閘門的生成方式，顯著降低建構時間與電路深度，同時維持相同的估測精度[4]。

本專題並非複製 LuGo 架構，而是參考其平行化設計理念，自行以數學方式直接計算出每一階段所需的 U 的指數次演化，再以 Qiskit 建構對應的 controlled-U gate，而非透過重複堆疊基本電路或仰賴超級電腦生成電路資料庫。此方式可於一般個人電腦環境下完成模擬，成功重現受控閘門重組與電路深度壓縮的效果，並保有準確的相位估測結果，顯示該策略具備實作於資源受限平台的潛力。為強化實作驗證，亦將標準 QPE 演算法移植至 FPGA 硬體平台，硬體系統以 Verilog 撰寫。結果顯示硬體實測結果與 Qiskit 模擬高度一致，成功驗證本設計於軟硬體平台均具正確性與穩定性。

此外，本研究更進一步嘗試將 QPE 演算法應用於無線通訊領域中，探討其於 MIMO（Multiple-Input Multiple-Output）系統的通道特徵值估計潛力。由於 MIMO 系統中的多根天線訊號會經由多條傳輸路徑抵達接收端，每條路徑會造

成不同的時延、衰減與相位偏移，進而構成一個通道矩陣 H。若未來能透過 QPE 準確估測該矩陣之主特徵值與相位資訊，將有助於實現更有效率的波束賦形 (beamforming) 與通道識別，為量子通訊特徵提取模組提供新穎的建構方式。因此，本研究的目的在於：

1. 以 Qiskit 實作並驗證標準 QPE 電路行為，理解相位估測流程與模擬結果；
2. 探討 QPE 演算法的運作機制與效能瓶頸；
3. 自行設計並實作改良式 QPE 電路架構；
4. 利用改良式 QPE 驗證量子方法在矩陣特徵值相位估計的可行性，作為量子通訊特徵提取模組雛形；
5. 設計並實現支援多種 PSK 模式的 FPGA 硬體平台；
6. 進行軟硬體結果比對與分析。

1. Quantum Phase Estimation 原理

量子相位估計（Quantum Phase Estimation, QPE）是量子計算中一項關鍵演算法，其目的是估算一個 unitary operator U 的本徵向量 $|\psi\rangle$ 所對應的本徵值中所隱含的相位 ϕ ，若特徵向量滿足

$$U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle \quad (1)$$

則 QPE 就可以透過一系列量子邏輯操作，將這個難以直接觀測的相位資訊，轉換為可由量子測量取得的二進位結果。核心概念在於將「複數特徵值中所隱藏的相位角」轉換為一組可測量的量子位元，藉由量子干涉與逆傅立葉轉換（IQFT）將其解碼為二進位表示。

2. QPE 電路結構與運作流程

QPE 的電路結構可分為五個主要步驟。以 n 個控制 qubit 為基礎說明：

2.1. 初始化控制暫存器與目標暫存器

第一暫存器(控制 qubit)：n 個控制 qubit，初始狀態為 $|0\rangle^{\otimes n}$。第二暫存器(目標 qubit)：設為特徵向量 $|\psi\rangle$，且滿足式 (1)，因此初始整體狀態為：

$$|\psi_0\rangle = |0\rangle^{\otimes n}|\psi\rangle \quad (2)$$

2.2. 對控制 qubit 套用 H gate

每個控制 qubit 轉為疊加態變成：

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n}|\psi\rangle \quad (3)$$

2.3. 施加 Controlled $-U^{2^j}$ 操作

當第 j 位 qubit 為 $|1\rangle$ 時，受控操作會將對第 j 位 qubit 施加不同次方的 U^{2^j} 運算：

$$U^{2^j} | \psi \rangle = e^{2\pi i 2^j \phi} | \psi \rangle \quad (4)$$

雖然目標 qubit (即特徵態$| \psi \rangle$) 本身狀態不變，但每次的受控操作都會將相位資訊累積至控制 qubit，形成整體的相位干涉。最終控制暫存器狀態變為：

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} (|0\rangle + e^{2\pi i 2^{n-1}\phi} |1\rangle) \otimes (|0\rangle + e^{2\pi i 2^{n-2}\phi} |1\rangle) \otimes \dots \otimes (|0\rangle + e^{2\pi i 2^0 \phi} |1\rangle) \otimes |\psi\rangle \quad (5)$$

乘積中每一個 qubit 都攜帶一段與 ϕ 相關的相位。總體乘積結構仍可進一步展開為一個對 Fourier 基底的線性組合如式(6)。每個 n－qubit 基底狀態 $| k \rangle$ 對應一個整數編號 k，並帶有特定的相位因子 $e^{2\pi i \phi k}$ 。因此，控制暫存器最終的量子狀態可簡化表示為：

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i \phi k} | k \rangle \otimes | \psi \rangle \quad (6)$$

2.4. 施加 Inverse QFT

已知 QFT 的作用:

$$| j \rangle \xrightarrow{QFT} \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i j k}{2^n}} | k \rangle \qquad (7)$$

接下來，對狀態 $|\psi_2\rangle$ 施加逆量子傅立葉轉換（Inverse QFT），並以式(6)的展開結果作為輸入表示形式，進一步將控制暫存器中所承載的相位資訊轉換為可觀測的二進位測量結果如式(8)：

$$|\psi_2\rangle \xrightarrow{IQFT} \frac{1}{2^n} \sum_{j=0}^{2^n-1} \sum_{k=0}^{2^n-1} e^{-\frac{2\pi i j k}{2^n}} e^{2\pi i \phi k} | j \rangle \otimes | \psi \rangle = \frac{1}{2^n} \sum_{j=0}^{2^n-1} \sum_{k=0}^{2^n-1} e^{-\frac{2\pi i k}{2^n}(j - 2^n \phi)} | j \rangle \otimes |\psi\rangle \quad (8)$$

而經過 Inverse QFT，原本以 k 為指標的傅立葉型態被還原為 j 為基底的整數態，其中 j 即為最終可測量的估測結果，期望近似 $\mathbf{2^n \phi}$。Inverse QFT 扮演著解碼器的角色，將控制暫存器中以相位編碼的訊息轉換為可觀測的二進位整數輸出，從而完成量子相位的估測。

2.5. 測量控制暫存器

由於 QPE 的量測結果並非直接得到相位 ϕ ，而是得到一個整數 j ，滿足下列近似關係：

$$j \approx 2^n \hat{\phi} \quad (9)$$

因此可反推出估計相位為：

$$\hat{\phi} = \frac{j}{2^n} \quad (10)$$

這裡的 j 即為 QPE 最終控制暫存器的量測結果。而理論上，當位元數 n 越大估測精度也越高，且誤差可被嚴格限制在：

$$|\phi - \hat{\phi}| < \frac{1}{2^{n+1}} \quad (11)$$

整體電路圖如圖 1.

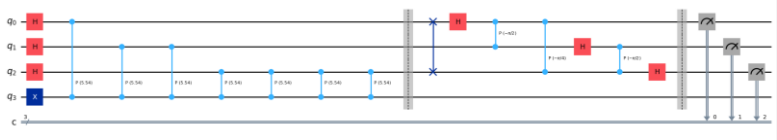

圖 1． 以 Qiskit 自行撰寫程式所生成之 QPE 電路圖（3 qubit 範例）

3. 傳統 QPE 電路設計瓶頸

雖然 QPE 演算法理論上具有優雅的架構與高度通用性，但其在實際電路實作上存在若干限制，尤其是步驟 **2.3 施加 $Controlled - U^{2^j}$ 操作的堆疊設計**，對電路深度與資源消耗造成極大負擔。

在傳統 QPE 設計中，為了實現對相位 ϕ 的高精度估計，需對每個 qubit 套用一次$Controlled - U^{2^k}$ 操作，這表示第 k 個 qubit 會受到 U 的 2^k 次方。若使用 Trotter 分解或直接由基本邏輯閘堆疊構成，將導致單一控制閘需重複數百至數千次，進而使整體電路深度呈指數增長。

提到的 Trotter 分解是一種常用的方法，用於將複雜的 Hamiltonian 演化近似拆解為多個可實作的子單位元操作。假設 H=A+B 則可透過以下公式進行近似：

$$e^{i(A+B)t} \approx \left(e^{\frac{iAt}{n}} e^{\frac{iBt}{n}}\right)^n \quad when\ H = A + B \quad (12)$$

當 $n \to \infty$ 時，此近似將趨近精確。這種方法常用於標準 QPE 中建構 $U^{2^k} = e^{iH2^k}$ 型態的受控閘電路。但隨著 k 增大，所需的 Trotter 步驟數會**指數成長**，導致電路變得極為冗長，圖 2 與圖 3 紅色框即為重複受控閘電路部分，這就是 QPE 實作困難的原因之一。此外，這些受控閘不僅深度高、數量多，也極度依賴精準的邏輯結構與穩定的量子閘操作。因此，如何降低受控單位元重複次數、縮短電路深度並提升生成效率，成為推動 QPE 實用化的關鍵挑戰，也正是 LuGo 架構提出的切入點。

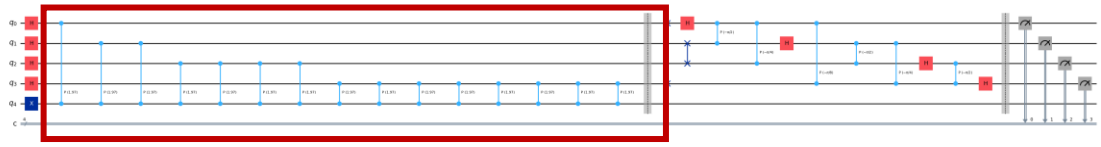

圖 2. 未優化 classical QPE on 4 qubits 電路圖

圖 3. 未優化 classical QPE on 5 qubits 電路圖

4. LuGo-QPE 之優化理念（平行生成、Peter–Weyl 分解）

為克服傳統 QPE 在受控單位元操作重複堆疊及電路深度快速增長的瓶頸，Lu 等人於 2025 年提出 LuGo（Lu's Optimized Generator for QPE）架構，設計一套支援平行化且具高擴展性的 QPE 電路生成流程，顯著改善電路深度與建構效率，並成功應用於如 Harrow–Hassidim–Lloyd（HHL）等需頻繁調用 QPE 的演算法中。LuGo 的核心優化策略可歸納如下：

4.1. 避免重複堆疊控制閘：

LuGo 並未使用傳統 QPE 中以重複方式實現的 $U^{2^i} = e^{iH2^i}$ 受控閘，轉而採用**單次生成所有必要次方的 unitary gate**。

4.2. 引入 Peter–Weyl 分解法：

結合 Peter–Weyl 理論，針對任意 unitary 矩陣進行分解，以較少邏輯閘數完成 operator 合成，取代傳統 Trotter 展開法所需的大量小步近似，有效減少資源消耗與電路誤差。

4.3. 利用超級電腦（Frontier）平行生成所有受控閘：

LuGo 架構將每個 U^{2^i} 的 controlled-unitary gate 視為獨立模組，透過 Python 的平行計算工具實作多執行緒設計。針對**超大型矩陣（26x26 ~ 29x29）**，由於每個 controlled-分解要耗掉大量記憶體和計算時間因此必須結合**平行化處理與超級電腦運算資源**，加速整體受控閘的構造過程，避免傳統 QPE 中需要逐一複製電路的過程，大幅降低運算時間與記憶體需求。

5.由 LuGo 啟發之程式架構與改良構想

在深入閱讀 LuGo 論文後，我觀察到其優化雖大幅改善 QPE 的電路深度與建構效率，但也存在部分限制，例如需依賴具幾百核心的超級電腦以支援大規模平行運算，且 Peter–Weyl 分解具備一定的數學背景門檻。基於此，我提出一套由 LuGo 啟發、針對一般開發環境設計的簡化版本。

5.1. Qiskit 自動分解：

使用 Qiskit 的 unitary() 方法將矩陣轉換為量子電路，由框架自動處理門分解。此方法會將任意 unitary 矩陣交由框架內建的分解器自動處理，展開成基本閘序列。雖然無需手動實作任何矩陣分解演算法，但 Qiskit 背後實際上使用了 **Cosine-Sine decomposition (CSD)**（適用於多 qubit）和 **KAK decomposition**（適用於 2 qubit）是實現任意 unitary 矩陣分解的常用方法。這兩種分解策略建立在李群表示論的數學基礎之上，將矩陣展開為局部旋轉與多 qubit 糾纏操作的組合，可視為 Peter–Weyl 理論在量子電路合成上的工程化應用。因此，雖然本程式在設計上「看似省略 Peter–Weyl 分解」，實際上 Qiskit 已隱式執行相關分解，使得矩陣直接轉換為量子電路成為可能。

5.2. 優勢

直接矩陣運算生成：將每個 U^{2^i} 由數值運算直接計算後，交由 Qiskit 自動分解為量子電路。

電路深度與執行時間的顯著優化：相較於標準 QPE，在小型 Toeplitz 矩陣測試中，電路深度減少超過 80%，運算時間提升數倍，顯示改良設計在電路生成效率上的實用價值。

便於誤差分析與驗證：程式先以經典演算法求解目標矩陣特徵值與特徵態，再以量子電路執行 QPE，兩者結果進行對比。此設計可量化 QPE 的估計誤差，為電路正確性提供強有力的驗證。

架構平行潛力：雖然目前為單執行緒設計，但每個 U^{2^i} 獨立生成的特性也允許後續導入 Python 多執行緒（如 multiprocessing 或 Ray）進行平行化，進一步加速大矩陣的電路建構。

無須依賴超級電腦：原始 LuGo 論文需依賴數百核心的超級電腦以支援大規模平行運算，而本程式改良版透過直接矩陣運算與 Qiskit 自動分解，於一般個人電腦即可執行 QPE 測試，降低開發門檻，提升開發環境適用性。

5.3. 侷限

依賴經典演算法求特徵態：目前版本以 NumPy 求解矩陣的特徵值與特徵態，作為 QPE 的輸入。但可能在真實量子硬體上不可行，限制了程式在「未知 Hamiltonian」場景的應用。

模擬限定於小矩陣：目前 Toeplitz 矩陣測試僅涵蓋 2×2 與 4×4 規模。對於大矩陣（如 16×16 以上）的電路生成與 QPE 準確性尚未評估，無法完全呈現 LuGo 優化在大型系統的效益。

參、 研究方法與實作流程

1. 硬體實作

將 QPE 搬移至 FPGA 平台，設計完成按鍵控制、Moore 狀態機、七段顯示等模組，並透過 LFSR 模擬雜訊環境進行容錯驗證。以 Top_BTN 為頂層模組，整合各子模組並負責狀態控制與顯示。整體架構如圖 4：

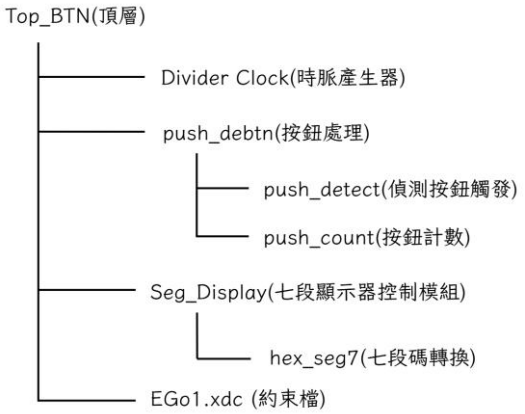

圖 4. 以 Verlilog 撰寫 FPGA 模組階層圖

主要架構選擇採用摩爾有限狀態機，輸出僅依賴當前狀態，而非輸入變化，因此輸出會在狀態轉移時（由時脈觸發）才更新。圖 5~7 為本系統所設計的狀態轉移條件與對應行為說明：

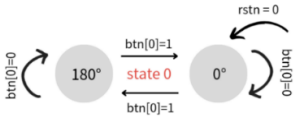

圖 5. BPSK 控制二狀態循環

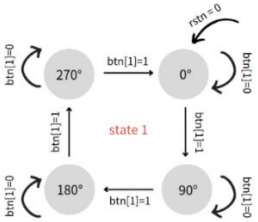

圖 6. QPSK 控制四狀態相位循環

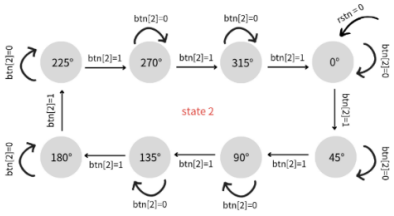

圖 7．8-PSK 控制八狀態切換

偽隨機相位誤差：模擬干擾條件

為了模擬實際通訊環境中的隨機相位擾動，本系統設計一組 4-bit 線性回饋暫存器（Linear Feedback Shift Register, LFSR），用於產生偽隨機數列。為了讓每次輸入的相位都有變化，在按下任一按鈕時，產生一個隨機偏移量。表一列出由 LFSR 所產生的 15 組偽隨機序列，並對應各自的 Offset 值。LFSR 順序循環，最終會回到 0001，形成一個封閉的偽隨機週期。

表 1. 生成的 15 組偽隨機數據順序

(bin)	(dec)	LFSR%10	Offset
0001	1	1	100000
0010	2	2	200000
0100	4	4	400000
1001	9	9	900000
0011	3	3	300000
0110	6	6	600000
1101	13	3	300000
1010	10	0	000000

(bin)	(dec)	LFSR%10	Offset
0101	5	5	500000
1011	11	1	100000
0111	7	7	700000
1111	15	5	500000
1110	14	4	400000
1100	12	2	200000
1000	8	8	800000
0001	1	1	100000

Divider_Clock 模組設計邏輯

在 FPGA 設計中，系統時脈(如 50MHz)往往遠高於周邊模組所需時脈頻率。為了讓按鍵偵測、七段顯示等模組能穩定運作，必須將主時脈「除頻」成較低的頻率。因此設計了一個多組輸出時脈的除頻器模組 Divider_Clock，產生不同頻率供各模組使用。

表 2.各模組使用的頻率

頻率	時脈名稱	使用模組
50MHz	sys_clk_in	Divider_Clock
1kHz	clkout_1kHZ	Seg_Display , Top_BTN
100Hz	clk_100	push_debtn , Top_BTN
10Hz	clk_10	push_debtn , push_detect , push_count

push_debtn 模組設計邏輯

處理按鈕輸入，將有雜訊（抖動）的原始按鈕訊號（btn），轉換成乾淨且穩定的單次脈衝（btn_r），讓後面狀態機可以穩定跳轉。

push_detect 模組設計邏輯

判斷是否有按下按鈕，並輸出觸發訊號。當偵測到 btn[i] 被按下時，模組會使用 **5-bit one-hot 編碼**（例如 5'b00010 表示按下 btn[1]）將結果輸出至 state 訊號。

push_count 模組設計邏輯

按鈕觸發後，會將對應的 One-Hot 編碼存在 pos，並透過計數器讓 btn_de 維持一段時間。本設計中引入時脈為 10Hz，設定 T05S＝5，代表輸出保持約 0.5 秒，達到按鈕去彈跳的效果。計時結束後會自動清除，等待下一次按鍵。

Seg_Display 模組設計邏輯

透過 1kHz 掃描時脈與 state 輪流切換顯示位，讓 8 位數看起來同時亮起。

hex_seg7 模組設計邏輯

將輸入的 5-bit 數字轉換為七段顯示器所需的 8-bit 控制格式。

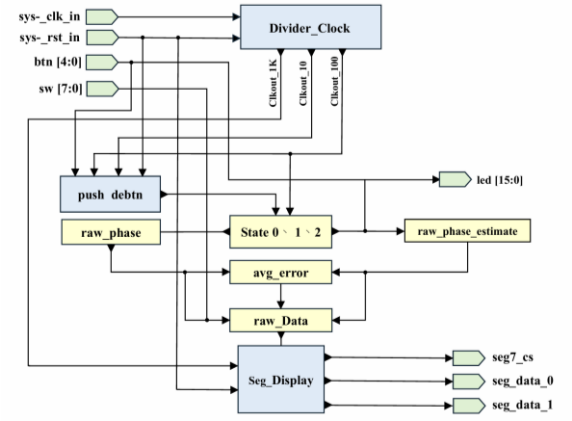

圖 8. Verilog 設計圖變數傳遞流程

2. 模擬實驗

2.1 模擬經典 QPE (因電路篇幅限制以 $qnum = 3$ 做舉例)

Step1: 初始化量子與經典暫存器

```
qnum = 3
q = QuantumRegister(qnum + 1, 'q')
c = ClassicalRegister(qnum, 'c')

circuit = QuantumCircuit(q,c)
```

圖 9. Step 1 (Qiskit)

額外多加 1 個 qubit (q[qnum])，作為應用控制單位 U 的「目標 qubit」。

Step2: 創建疊加態並設定目標 qubit

```
for i in range(qnum):
    circuit.h(q[i])
circuit.x(q[qnum]) # Flips Q[qnum] to 1
```

圖 10. Step 2(Qiskit)

將前 3 個 qubit 放入均勻疊加態，為 QPE 提供干涉基礎。並將最後一個 qubit（目標 qubit）初始化為狀態 |1⟩，準備後續受控相位旋轉作用（Controlled-U 操作的目標）。

Step3: 施加受控相位操作

```
actual_phase = 0.754
angle = 2*pi*actual_phase

for i in range(qnum):
    for _ in range(2 ** i):
        circuit.cp(angle, q[i], q[qnum])
```

圖 11.Step 3 (Qiskit)

每個計數 qubit 根據其位置 i，施加 2^i 次的受控相位門 $CP = e^{i2\pi\theta}$ 模擬 U^{2^i}的效果。

圖 12.Step 1-3 電路圖

Step4: Inverse QFT

```
for j in range(qnum // 2):   # Bit-reverse swap
    circuit.swap(q[j], q[qnum - j - 1])
for j in range(qnum):  # Inverse QFT gates
    circuit.h(q[j])
    for k in range(j + 1, qnum):
        circuit.cp(-np.pi / (2 ** (k - j)), q[j], q[k])
```

圖 13.Step 4 (Qiskit)

先進行 bit-reversal swap，將 qubit 順序對調以符合 QFT 格式。並對每個 qubit 進行 Hadamard 閘與受控相位旋轉 $CP(-\frac{\pi}{2^k})$ ，這兩步驟就是 IQFT，把藏在量子狀態中的相位資訊還原出來。

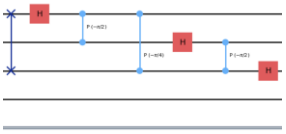

圖 14. IQFT 電路圖(3-qubit)

Step5:測量

```
for i in range(qnum):
    circuit.measure(q[i], c[i])
```

圖 15. Step 5 (Qiskit)

量測後原本以疊加與干涉方式儲存的相位資訊，會**塌陷成一組** bit pattern 結果。

Step6:觀察量子狀態（statevector）

以 **Bloch 球**視覺化每個 qubit 的量子態

```
sim = Aer.get_backend("aer_simulator")
circuit_init = circuit.copy()
circuit_init.save_statevector()
statevector = sim.run(circuit_init).result().get_statevector()
plot_bloch_multivector(statevector)
```

圖 16. Step 6 (Qiskit)

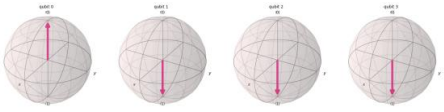

圖 17. Bloch 球顯示測量結果

Step7：結果解析與相位估計

```
a = counts.most_frequent()
bin_a = int(a,2) # Converts the binary value to an integer
phase = bin_a/(2**qnum) # The calculation used to estimate the phase
end_time = time.time()
print(f"\n[Classical QPE]")
print('qubit number is: ',qnum)
print(f'Most frequent measurement: ',a)
print(f"Estimated phase :{phase:.9f}")
print(f"True phase :     {actual_phase:.9f}")
print(f"Absolute error:  {abs(phase - actual_phase):.9f}")
print("circuit Depth:", circuit.depth())
print(f"Total time:{end_time - start_time:.6f} 秒")
```

圖 18． 測量計算結果

將實際相位值與估測結果一併輸出進行比較。從模擬結果中選取出現次數最多的二進位字串量測結果 a ，該 bit 結果代表 QPE 對實際相位的最佳估計。並利用 python 中 int() 函式的標準用法將字串 a 當作二進位數字轉換為十進位整數，例如 "101" → 5。最後根據式 10 的整數近似，因此反推 $\hat{\phi}$ 可用：

$$估計相位\ \hat{\phi} = \frac{int(a,2)}{2^{qnum}} \quad (13)$$

2.2 非 Trotter、似 Peter–Weyl：以數值矩陣計算建構 QPE 電路

儘管 LuGo 原始設計中採用如 ray 等工具平行化生成 gate，並透過 Peter–Weyl 分解法壓縮電路，本專題考量實驗規模、Qiskit 環境限制及簡潔性，未納入上述技術實作。相較於傳統 QPE 常見 Trotter 展開法，以及 LuGo 採用的 Peter–Weyl 分解方法，本研究直接利用 Python 的矩陣指數運算搭配 unitary() 方法合成量子閘，實現操作。儘管如此，Qiskit 在 unitary() 背後仍隱含使用 Cosine-Sine 分解（CSD）或 KAK 分解進行最佳化，本質上承襲了 LuGo 所強調的 Peter–Weyl 工程化精神。

同時，在應用層面上，本研究亦嘗試將 QPE 演算法模擬應用於 MIMO 通訊系統的通道估計場景。藉由 QPE 估計通訊系統的通道 H 的特徵值，可協助辨識主要傳輸通道，進而導引波束調整方向、降低能量損耗。模擬流程如下：

Step1：建立 Hermitian Toeplitz 矩陣

```
def generate_tridiagonal_toeplitz(n):
    A = 2 * np.eye(n)
    for i in range(n - 1):
        A[i, i + 1] = -1
        A[i + 1, i] = -1
    return A
```

圖 19. 三對角 Toeplitz 矩陣產生函數

本研究使用 np.eye(n) 產生 n×n 的單位矩陣，並進一步調整其上下對角線為 -1，對角線為 2，以形成對稱三對角的 Toeplitz 結構便於分析與模擬。然而，在真實應用中，通道矩陣 H 不一定為 Hermitian，因此將需先透過數學轉換（例如 Hermitian 擴展或對稱化處理）使其符合量子相位估計（QPE）所要求的 Hermitian 性質。

在本模擬階段，則選用已知為 Hermitian 的 Toeplitz 結構作為 Hamiltonian 的代表範例，以利驗證演算法的可行性與穩定性。以 n = 4 為例，創造出的矩陣 A 為：

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \quad (14)$$

此矩陣為實對稱矩陣，亦即其共軛轉置等於自身，符合 Hermitian 的定義，因此適合作為 QPE 中的 Hamiltonian 來源。

Step2：生成 unitary operator

```
L = generate_tridiagonal_toeplitz(a)
U = expm(1j * 2 * pi * actual_phase * L)
```

圖 20. 以矩陣指數運算生成 unitary operator U

在 QPE 中，我們的目標是估計某個 unitary operator U 的本徵值 λ，其形式為 $\lambda = e^{2\pi i\phi}$，其中 ϕ 為要估計的相位，而對應本徵向量為 $|\psi\rangle$。兩者滿足關係式 $U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle$。為了模擬此行為，我們需要構造一個特定結構的 unitary operator $U = e^{2\pi i\phi L}$，其中 L 為一個 Hermitian 的 Toeplitz 矩陣，其本徵值記為 $\lambda_k{}^{(L)}$。由於 Hermitian 矩陣皆可對角化為 $L = V\Lambda V^{-1}$，則有 $U = e^{2\pi i\phi L} = Ve^{2\pi i\phi\Lambda}V^{-1}$。因此，對於每個本徵向量 $|\psi_k\rangle$，其對應的本徵值為：$\lambda_k = e^{2\pi i\phi\lambda_k{}^{(L)}}$，使用 scipy.linalg.expm() 函數完成上述矩陣指數運算，將 U 實作為 Qiskit 可接受的 unitary gate，以供後續 QPE 電路使用。

Step3：計算 eigenstate 與 phase

為了驗證 QPE 電路的估測能力，在模擬前先對 unitary operator U 進行本徵分解，透過 NumPy 的 linalg.eig()函數，同時取得本徵值 λ_k 與本徵向量 $|\psi_k\rangle$ 作為 QPE 的輸入狀態。

```
eigvals, eigvecs = np.linalg.eig(U)
print("--- Eigenvalues and Phases ---")
for i, eigval in enumerate(eigvals):
    phase = np.angle(eigval) / (2 * np.pi)
    if phase < 0:
        phase += 1
    print(f"Eigenvalue {i}: {eigval:.4f}, Phase: {phase:.6f}")
```

圖 21. 擷取 unitary operator 本徵值並換算相位

Step4： 建立改良式電路

在改良式 QPE 中，最關鍵的優化環節即為 controlled-U^{2^k} 閘的建構方式。本實作中**直接以矩陣運算產生每個所需次方的** unitary operator，取代傳統方法中重複堆疊單位閘所造成的深度增長問題。

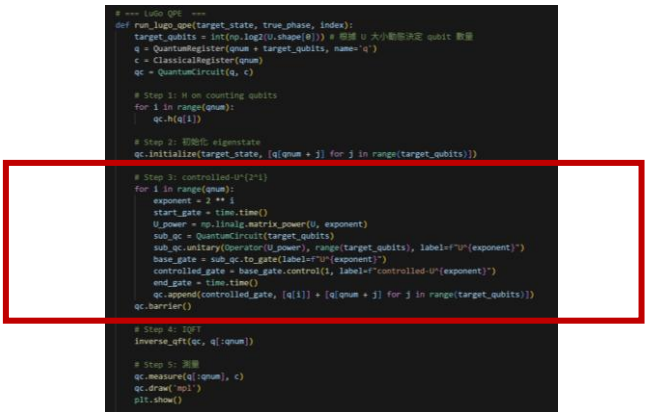

圖 22. 實作之改良式 QPE 主程式架構

使用 np.linalg 直接計算第 i 位計數 qubit 所需的 unitary 操作次方，再將其轉換為 Qiskit 可接受的 gate 物件。接著利用.control(1) 將原本的 unitary gate 封裝為受控操作，形成 controlled-U^{2^k} 結構。此方法有效取代傳統「一層一層堆疊 U 的方法」，大幅降低電路深度，並保留每個次方操作的結構清晰性。

肆、 實驗結果與分析

1. 硬體實作

雖然系統具備支援 BPSK 與 QPSK 模式的能力，此說明書將聚焦於 8-PSK 模式，說明其操作流程及相關顯示內容。

作品連結: https://www.youtube.com/watch?v=gF_aaEVSi24

1.1 進行相位顯示切換

按下 S2 按鈕（每次按下時右側 LED 會閃爍），將模式設為 8-PSK 並切換至第一相位。

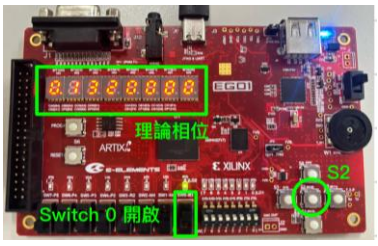

圖 23. 8-PSK 第一跳轉理論相位

圖 24. 8-PSK 第一狀態估計相位

圖 23 顯示理論值為 0.125 加上 0.007 (LFSR 產生的隨機干擾)，而當 Switch0 開啟（向上） 時，七段顯示器顯示理論相位值，如 0.1320000。當 Switch0 關閉（向下） 時，顯示量測後的估計相位值，如 0.1328125，並搭配使用 Qiskit 觀察執行結果數據進行比對。

1.2 記錄相位誤差

按下 S4 按鈕可將系統記錄此組理論與估計相位的誤差。

```
[Classical QPE]
qubit number is:  8
Most frequent measurement:  00100010
Estimated phase :0.132812500
True phase :     0.132000000
Absolute error:  0.000812500
circuit Depth: 273
Total time：5.369903 秒
```

圖 25．使用 Qiskit 執行結果

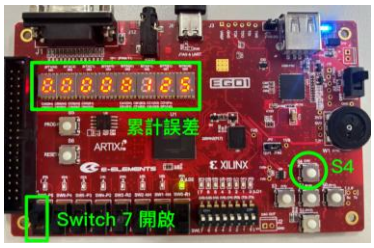

圖 26. 8-PSK 第一狀態理論&估計誤差

開啟最左側的 Switch7，畫面會顯示累積平均誤差值，如 0.0008125，代表目前估計與理論值的誤差統計。

1.3 切換至下一個相位

再次按下 S2，系統跳轉至下一組相位。

圖 27. 8-PSK 第二跳轉理論相位

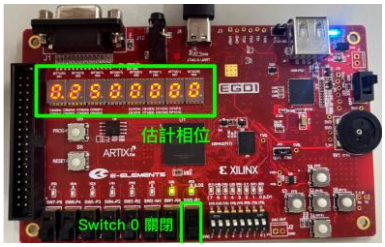

圖 28．8-PSK 第二狀態估計相位

畫面顯示 0.251 為新一組的理論相位值。同時 LED 也會根據相位索引累積亮燈數量（亮兩顆代表索引為 2）。關閉 Switch0 後，畫面顯示估計值 0.25，並按下 S4 按鈕使系統記憶誤差。

1.4 顯示估計值與累積平均誤差

再次開啟 Switch7，畫面顯示更新後的累積平均誤差值 0.0009062，並搭配使用 Qiskit 觀察執行結果數據進行比對。

```
[Classical QPE]
qubit number is:  8
Most frequent measurement:  01000000
Estimated phase :0.250000000
True phase :     0.251000000
Absolute error:  0.001000000
circuit Depth: 273
Total time：5.856911 秒
```

圖 29．使用 Qiskit 執行結果

圖 30．8-PSK 第二狀態理論&累積平均誤差

計算方式如下，並取至小數點後七位：

$$AvgError = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{\phi}_i - \phi_i\right| \qquad (15)$$

其中 $\hat{\phi}_i$ 為第 i 次的估計相位值，ϕ_i 為第 i 次的理論相位值，N 為累積次數（按下 S4 的次數）。因此此時累積平均誤差為：

$$\frac{1}{2}(0.0008125 + 0.001) = 0.00090625 \quad (16)$$

2. 模擬實驗

2.1 模擬經典 QPE

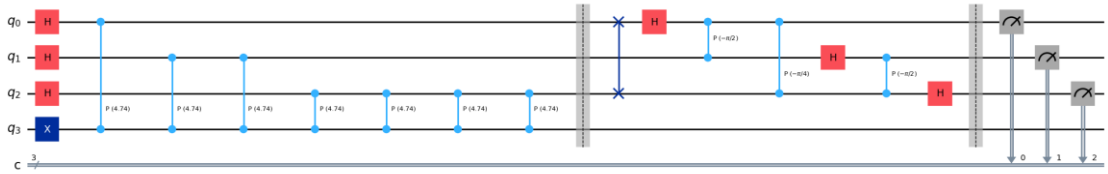

圖 31．以 Qiskit 自行撰寫程式所生成之 QPE 電路圖（3 qubit 範例）

主要分成三階段，在 QPE 編碼階段中的每一層 CP(4.74)表示受控的 $e^{i2\pi\theta}$ 操作，由於預估相位設為 0.754，因此每個 controlled qubit 上的 cp(angle, q[i], q[qnum]) 就會是：

$$angle = 2\pi \times actual\ phase\ = 2\pi \times 0.754 = 4.7375 \quad (17)$$

Bloch 球圖觀察 qubit 狀態

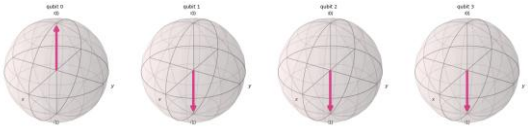

圖 32．Bloch 球圖觀測 QPE on 3 qubit

量測結果直方圖：經過 *Step6* 的 8192 次模擬後，各種 3-bit 量測結果出現的次數統計分布如圖 33。

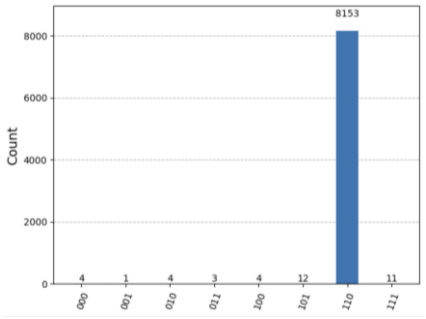

圖 33．量測結果分布

```
[Classical QPE]
qubit number is:  3
Most frequent measurement:  110
Estimated phase :0.750000000
True phase :      0.754000000
Absolute error:  0.004000000
circuit Depth: 15
Total time：1.584598 秒
```

圖 34．量測結果數據顯示

圖 33 可看出最大值為 110，出現 8153 次，佔總次數的 99.5%。而圖 34 顯示在本次實驗中，使用 3 個 qubit 對應的相位估計值最常見輸出為 110。而原本真實相位 = 0.754，非常接近 0.75，誤差僅 0.004，若使用的 qubit 數越多，結果也將越精確。

為了更全面評估 QPE 演算法在不同 qubit 數下的表現，進一步針對整個相位範圍進行模擬。具體而言，將 target phase 自 0 遞增至 0.99，間隔 0.01 共 100 點，並對每個相位點進行 1000 次量測，統計出最常見的量測結果作為該點的估計值。

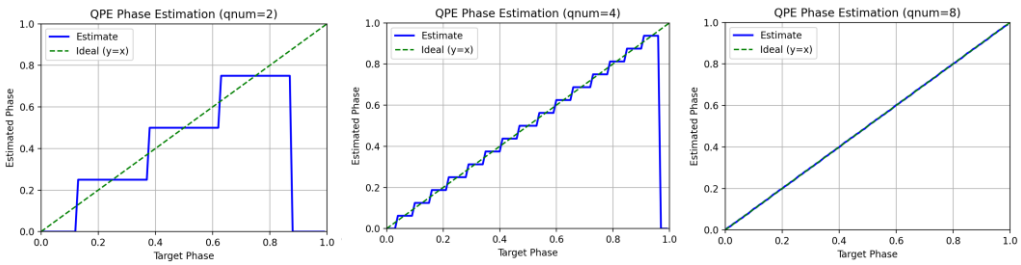

圖 35-37．左至右分別為 qnum＝2、3、4 之 QPE 模擬與理論比對結果

圖 35-37 即展示三種 qubit 數設定下的估計結果。藍色線條為實際估計值，綠色虛線為理想估計線 y＝x。可觀察到，隨著 qubit 數增加，估計結果越接近理想線，表示相位估計越精準。

2.2 自行設計之改良式 QPE

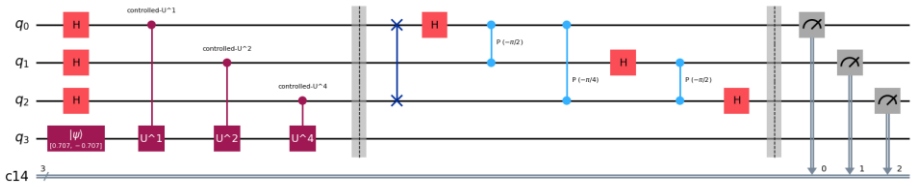

圖 38．LuGo-QPE on 3 qubit 電路圖

2.3 兩者電路深度比較

表 3. QPE 電路深度比較與改良法之降低比例

計數 qubit 數	Classical QPE	改良式 QPE	深度降低比(%)
2	9	8	11.1
3	15	11	26.7
4	25	14	44.0
5	43	17	60.5
6	77	20	74.0
7	143	23	83.9
8	273	26	90.5
9	531	29	94.5
10	1045	32	96.9

如表 3 所示，本研究所設計的改良式 QPE 電路（參考 LuGo 核心概念並經由簡化重構）在電路深度方面，較傳統 QPE 展現出顯著優勢。當計數 qubit 為 2 時，深度下降 11.1%；而隨 qubit 數增加，深度降低比例迅速提升，10 qubit 時已減少達 96.9%。本設計有效抑制了原始 QPE 電路中深度隨 qubit 數指數成長的問題，展現良好的可擴展性與實用潛力。

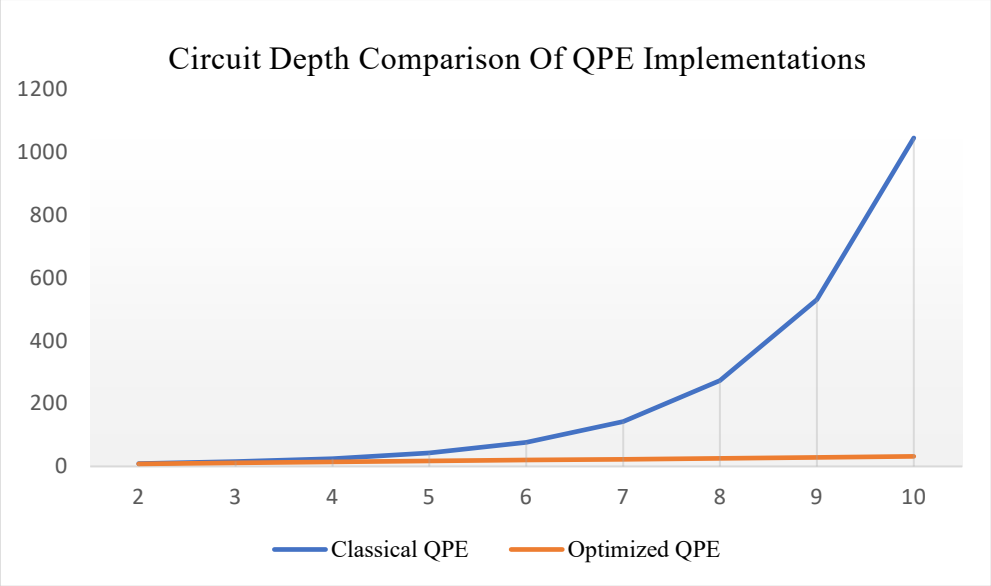

圖 39. QPE 電路深度比較圖

從圖 39 中更能看出兩者深度趨勢差距，傳統 QPE 電路所需的受控單位元操作數量呈指數成長，使得整體電路深度亦隨 qubit 數以指數級增加，而改良式 QPE 電路深度則隨 qubit 數**呈線性增長**，展現出優異的擴展性與實作效能。即使在 10 qubit 下，電路深度亦僅為傳統方法的不到 1/30，代表能更簡潔運行於當前量子硬體條件下。

2.4 隨 qubit 數量增加所需執行時間變化

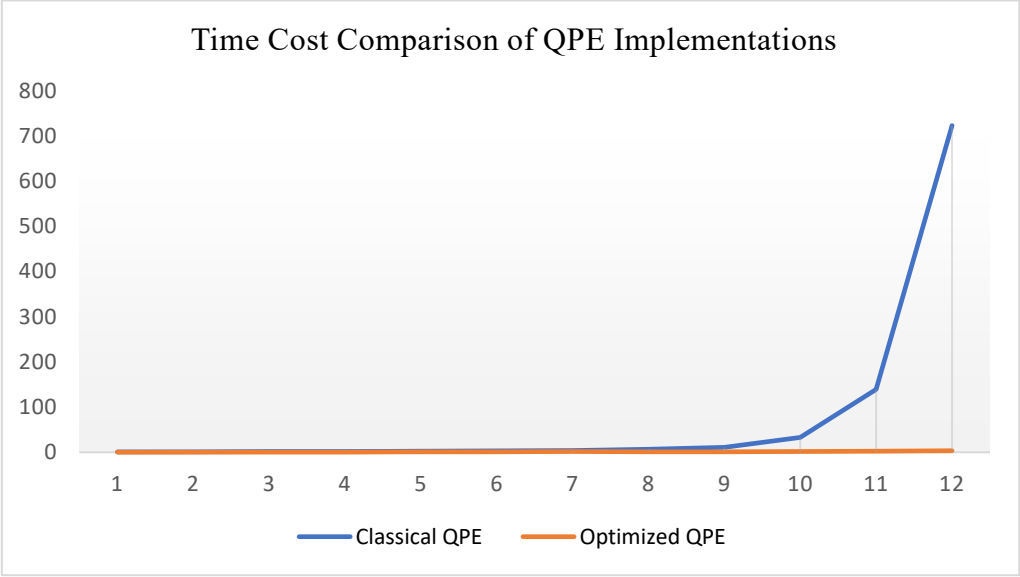

圖 40. QPE 演算法於不同 qubit 數下之執行時間比較圖

圖 40 比較傳統 QPE 與本研究所提出之改良式 QPE 實作，在不同 qubit 數下所需的執行時間（單位：秒）。可觀察到，傳統 QPE 執行時間在 qubit 數達 9 以上時呈現**指數型**爆增，而改良式則維持**近乎線性增長**，12 qubit 時執行

時間僅約為 2 秒，對比傳統方法所需超過 700 秒，顯示極高的效率與擴展性。(各 qubit 數下兩種實作的執行時間數據詳見附錄一）

2.5 比對部分數據截圖紀錄_

表四. 詳細數據紀錄表

[Classical QPE] qubit number is: 3 Most frequent measurement: 011 Estimated phase :0.375000000 True phase : 0.314000000 Absolute error: 0.061000000 circuit Depth: 15 Total time：1.532645 秒	[Classical QPE] qubit number is: 8 Most frequent measurement: 01010000 Estimated phase :0.312500000 True phase : 0.314000000 Absolute error: 0.001500000 circuit Depth: 273 Total time：6.435761 秒	[Classical QPE] qubit number is: 12 Most frequent measurement: 010100000110 Estimated phase :0.313964844 True phase : 0.314000000 Absolute error: 0.000035156 circuit Depth: 4121 Total time：722.615077 秒
[LuGo-inspired QPE] qubit number is: 3 Most frequent measurement: 011 Estimated phase: 0.375000000 True phase: 0.314000000 Absolute error: 0.061000000 circuit Depth: 11 Total time：0.576544 秒	[LuGo-inspired QPE] qubit number is: 8 Most frequent measurement: 01010000 Estimated phase: 0.312500000 True phase: 0.314000000 Absolute error: 0.001500000 circuit Depth: 26 Total time：1.301227 秒	[LuGo-inspired QPE] qubit number is: 12 Most frequent measurement: 011100000010 Estimated phase: 0.437988281 True phase: 0.438000000 Absolute error: 0.000011719 circuit Depth: 38 Total time：3.436391 秒

伍、 創新點與貢獻

本專題從通訊系統中相位估計的基本需求出發，深入探討了量子相位估計（Quantum Phase Estimation, QPE）之演算法原理與實作流程。透過 Qiskit 平台建構完整 QPE 電路，並建構出更電路深度優化的相位估測效果，並進一步於 FPGA 平台實作可視化相位估測系統，使原本抽象的量子演算法得以具象化與操作化。具體創新與貢獻如下：

1. 自行設計具備 LuGo 精神之簡化 QPE 架構，適用於低資源平台

本研究獨立以 Qiskit 自行實作整體 QPE 電路構建流程，並僅參考 LuGo 所提出的「減少 $controlled - U^{2^k} gate$ 閘堆疊」之設計概念，作為發想起點。與 LuGo 電路優化不同，本研究改以矩陣指數運算直接生成 $U^{2^k} gate$ 的方式並搭配 Qiskit 內建的分解器將矩陣轉換為量子電路，進行 QPE 中關鍵受控閘之建構，可重現性高。

即使未搭配超級電腦等高效能運算資源，仍成功重現類似的電路深度壓縮效果，證明該簡化設計策略具備良好的模擬穩定性，未來有機會應用於 NISQ 時期小型量子裝置及教育場景，展現量子軟體開發的潛在價值。

2. 電路深度量化分析

本專題比較不同計數 qubit 數下之 Classical QPE 與改良式 QPE，顯示後者在 10 qubit 下仍僅需約 3% 的深度，節省比高達 96.9%，證實本方法在電路資源效率上的突破，對實際應用極具參考價值。

3. 整合 FPGA 實作與雜訊容錯驗證

以 Verilog 設計 QPE 邏輯電路，整合除頻器、Moore 狀態機與七段顯示模組，並加入 LFSR 模擬隨機相位擾動，觀察輸出穩定性，結果顯示 FPGA 輸出與模擬一致，驗證此設計具備跨平台可移植性與實用性。

4. 跨領域整合與概念驗證平台

雖本設計尚未直接應用於實際無線通訊系統，但已作為一套概念驗證平台，成功展現 QPE 在未來通訊接收端設計、相位解調與誤差分析等環節的潛在應用價值。整體設計結合了**量子計算**、**無線通訊原理**與**數位邏輯**設計三個面向，實踐「跨領域整合與工程轉譯」的研發精神。

陸、 量子科技展望

量子科技被視為繼蒸汽、電力與數位革命之後的下一場科技浪潮，其核心價值在於運算架構的根本改變，從經典比特進入量子位元的並行世界。特別是在通訊、密碼學、材料科學與機器學習等領域，量子演算法提供了前所未見的效能突破。本研究聚焦於量子相位估計（QPE）演算法的電路實作與應用潛力，實證其在「相位測量」與「跨域整合」上的技術優勢。

透過本作品的實作經驗，我體認到 QPE 並非僅存在於教科書與理論模型中，而是可以透過模擬與硬體落實轉化為具體的應用原型。未來，隨著量子演算法的可行性將逐步從模擬走向部署，而 QPE 所代表的精確特徵值估算技術，將可望應用於通訊解調器與多種信號處理場景中，成為推動量子科技與工程實務接軌的關鍵橋樑。

柒、 心得與對量子科技之觀點

量子科技對我而言曾是未知且遙遠的領域，但透過本次專題參與，我深刻體會到量子演算法也能透過明確的邏輯與方法進行建構與驗證。在專題過程中，我不僅觀察並分析現有設計的優缺點，更嘗試提出改良策略，使 QPE 的電路結構能在低資源環境中有效運作。本研究以技術實作視角切入，探索量子演算法如何應用於通訊領域的實際問題，並透過程式編寫與電路設計，將抽象公式轉化為可驗證、可視化、可解釋的技術實體，進一步深化我對量子科技的理解。

回顧整體專題歷程，我從最初的理論學習、模擬驗證到硬體建構，歷經反覆錯誤與修正，也逐步理解「電路深度」、「量子閘複雜度」與「演算法平台適應性」等關鍵量子工程指標的實務意義。儘管目前的 NISQ 裝置仍存在物理層面限制，但這不影響我們從應用端出發，透過模擬與電路轉譯累積經驗。對我而言，量子科技不再只是科幻或研究室內的專利，而是一個有潛力真正影響產業未來的實用技術，值得我們持續投入與探索。

捌、 參考文獻

[1] J. G. Proakis and M. Salehi, *Digital Communications*, 5th ed. New York, NY, USA: McGraw-Hill, 2008.

[2] J. M. Kahn and K.-P. Ho, "Spectral efficiency limits and modulation/detection techniques for DWDM systems," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 10, no. 2, pp. 259–272, Mar./Apr. 2004.

[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[4] C. Lu, M. G. Meena, and K. C. Gottiparthi, "LuGo: An Enhanced Quantum Phase Estimation Implementation," *Preprint*, Oak Ridge National Laboratory, Mar. 20, 2025.

[5] Qiskit Community, "Quantum Phase Estimation," *Qiskit Textbook*, GitHub, [Online]. Available on, https://github.com/qiskit-community/qiskit-textbook/blob/main/content/ch-algorithms/quantum-phase-estimation.ipynb

[6] IBM Quantum, "PhaseEstimation class — Qiskit documentation," *Qiskit API Reference*, [Online]. Available on, https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.PhaseEstimation

[7] J.-Y. Pan, *Lectures on Toeplitz Matrices*, Dept. of Mathematics, East China Normal University, [Online]. Available on, https://math.ecnu.edu.cn/~jypan/Research/papers/lect_Toeplitz.pdf

[8] C. C. Jou, "特殊矩陣 （九）：Hermitian 矩陣," *CCJou's Blog*, Jan. 5, 2010. [Online]. Available on, https://ccjou.wordpress.com/2010/01/05/%E7%89%B9%E6%AE%8A%E7%9F%A9%E9%99%A3-%E4%B9%9D%EF%BC%9A-hermitian-%E7%9F%A9%E9%99%A3/

[9] Qulacs Dojo, "4.2 Trotter Decomposition," *Qulacs Documentation*, [Online]. Available on, https://dojo.qulacs.org/en/latest/notebooks/4.2_trotter_decomposition.html

[10] ICTS, "Chapter 2: Quantum Simulation and Trotter Decomposition," *International Centre for Theoretical Sciences (ICTS)*, [Online]. Available on, https://www.icts.res.in/sites/default/files/seminar%20doc%20files/Chapter2.pdf

[11] R. Yalovetzky, P. Minssen, D. Herman, and M. Pistoia, "Solving linear systems on quantum hardware with hybrid HHL++," *Scientific Reports*, vol. 14, Article no. 20610, Sep. 2024. [Online]. Available on, https://www.nature.com/articles/s41598-024-69077-0

[12]IBM Quantum, "qiskit.synthesis.TwoQubitBasisDecomposer," IBM Quantum Documentation. [Online]. Available on, https://quantum.cloud.ibm.com/docs/en/api/qiskit/qiskit.synthesis.TwoQubitBasisDecomposer

玖、 附錄

附錄一: Classical QPE 與改良式 QPE 實作之執行時間比較表（單位：秒）

計數 qubit 數	Classical QPE	改良式 QPE
1	0.891648	0.261973
2	1.175843	0.434301
3	1.532645	0.576544
4	1.980628	0.532486
5	2.121678	0.738268
6	3.476827	1.014848
7	4.193645	1.416190
8	6.435761	1.301227
9	10.803946	1.333366
10	32.584824	1.691406
11	139.14111	2.575734
12	722.615077	3.436391

附錄二 ： FPGA schematic design