國立台北大學通訊工程學系專題報告

基於 FPGA 的量子相位估計應用於 PSK 信號測量與分析

FPGA-Based Quantum Phase Estimation for PSK Signal Measurement

411186028 鐘婉庭

指導教授：沈瑞欽 教授

執行期間: 2024 年 9 月 — 2025 年 6 月

壹、 摘要

本專題首先以 Python 環境中的 Qiskit 套件建立量子相位估計（Quantum Phase Estimation, QPE）模型進行模擬驗證，完成電路設計、模擬執行與理論驗證，確認演算法正確性後，進一步實現實體化應用，於 FPGA（Artix-7）平台上實現對應硬體架構。系統支援多種 PSK 調變模式(BPSK、QPSK、8-PSK)，可透過按鈕切換，七段顯示器顯示原始相位、估計值與平均累積誤差，LED 則指示相位索引。硬體以 Verilog 撰寫，整合除頻器、按鍵處理、Moore 狀態機、七段顯示控制器等模組，完成輸入控制與輸出顯示設計。其中，透過 LFSR 模擬隨機相位誤差，可測試系統在雜訊擾動下的容錯能力與估計穩定性。實驗結果與 Qiskit 模擬高度一致，驗證系統設計具正確性與穩定性。

關鍵詞: 量子相位估計、Qiskit 套件、Python、誤差累積、FPGA、Moore 狀態機、相位偏移鍵控

貳、 簡介

計畫背景

隨著通訊技術邁向低功耗、高安全性與極限效率的發展趨勢，例如 6G、衛星通訊與量子通訊等應用場景，訊號傳輸所面臨的條件日益嚴苛。系統必須克服長距離、低功率甚至單光子等級的物理限制，同時兼顧資料的保密性與可傳輸性[1]。為了因應這些挑戰，選擇合適的訊號編碼與調變方式成為關鍵課題。

在各種調變技術中，相位調變（Phase Shift Keying, PSK）具備顯著的頻譜效率與抗雜訊能力，特別適用於低功率與長距離的通訊環境。因其訊號能以固定相位點（constellation points）對應不同位元組合，有效提升單位時間內的資料傳輸量[1]。例如 QPSK 可透過 4 個相位對應 2 個位元，8-PSK 則對應 3 個位元，進一步提升頻譜效率[2]。

此外，相較於振幅編碼，相位訊號在傳輸過程中較不易受到功率衰減與通道干擾影響[2]，在 Rayleigh 或 Rician 衰落通道中，PSK 的錯誤率表現比振幅調變更穩定[3]。因此在光通

訊與量子通訊中扮演越來越重要的角色。然而，經典的 PSK 接收器通常依賴混頻與載波同步等機制進行解調，當訊號功率下降時，載波相位同步器更容易產生誤差[2]，導致整體解碼效能下降。以 BPSK 為例，其低功率下的錯誤率為：

$$P_b = Q\sqrt{2 \cdot \frac{E_b}{N_0}}$$

當位元能量 E_b 對雜訊功率 N_0 過低時，錯誤率趨近 0.5 幾乎等同隨機猜測[2]。此外，傳統接收器在處理極低功率甚至單光子訊號時亦面臨技術限制[4]，限制了它在弱訊號環境中的應用。

為了解決上述困境，量子相位估計提供了新的解碼思維。不依賴信號強度，而是透過量子態的疊加、受控么正運算與反向量子傅立葉轉換（inverse QFT），將不可觀測的相位資訊轉換為可測量的二進位輸出，已廣泛應用於 Shor's Algorithm、Hamiltonian Simulation 等量子演算法中，作為估算量子系統特徵值的核心技術[5]，展現其在高效估測相位上的潛力。

基於上述背景，本專題以 QPE 為核心概念，分別在軟體與硬體層面進行實作與驗證。在軟體部分，利用 Qiskit 建立真實的 QPE 量子電路，透過量子閘操作與逆量子傅立葉轉換，模擬相位編碼還原特徵值的完整流程，並與理論推導比對結果一致性。在硬體部分，則以 QPE 的運作邏輯為設計靈感，建構一套近似「量子式接收端」的概念架構。該架構模擬 PSK 調變輸入條件，支援 BPSK、QPSK、8-PSK 等多模式輸入控制，透過 Verilog HDL 與 FPGA 平台進行狀態控制與輸出顯示，並以七段顯示器呈現估測相位與誤差，提供一個近似量子估測行為的系統操作體驗。雖然 FPGA 端不具備真正的量子運算功能，但透過整合 Qiskit 模擬與硬體輸出比對，可作為一種具啟發性的跨領域應用驗證，展現量子估測技術於通訊場景中的潛在價值。

理論介紹

在進一步探討 QPE 演算法之前，需先理解 QFT 的原理與結構 [6][7]。

QFT 與經典 DFT 的比較

量子傅立葉轉換（QFT）是經典離散傅立葉轉換 (Discrete Fourier Transform , DFT) 在量子計算中的對應形式。不同於經典 DFT 將數列轉換為頻域， QFT 會將輸入量子態轉換為一組**帶有相位因子之疊加態**。經典 DFT 定義如下：

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{\frac{2\pi i j k}{N}} \quad (1)$$

其中：

x_j：輸入資料的第 j 項；

y_k：轉換後第 k 項；

N：資料總長度；

對應到量子情境， QFT 將每個基底狀態 $|j\rangle$ 映射成帶有相位因子的疊加態 $|k\rangle$。給定一個 n qubit 的量子狀態 $|j\rangle$，資料總長度為 2^n，其 QFT 定義為：

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

k 是從 0 到 $2^n - 1$的整數，用於表示所有可能的 n qubit 的基底狀態 $|k\rangle$。QFT 可進一步展開為下列乘積形式：

$$|j\rangle \xrightarrow{QFT} \frac{1}{\sqrt{2^n}} \otimes_{l=1}^{n} \left[|0\rangle + e^{2\pi i j 2^{-l}}|1\rangle\right] \quad (3)$$

表示 QFT 結果為 n 個 qubit 的張量乘積，每個 qubit 帶有**自己對應的相位**且彼此之間**不具糾纏**（**entanglement**），因此可以逐個施加量子閘操作實現。最後可以得到

$$\frac{1}{\sqrt{2^n}}(|0\rangle + e^{2\pi i 0.j_n}|1\rangle) \otimes (|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle)$$

$$\ldots (|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}|1\rangle) \quad (4)$$

以三個 qubit 為例，QFT 結果為：

$$|j_1 j_2 j_3\rangle \xrightarrow{QFT}$$

$$\frac{1}{2\sqrt{2}}(|0\rangle + e^{2\pi i(0.j_3)}|1\rangle) \otimes (|0\rangle + e^{2\pi i(0.j_2 j_1)}|1\rangle)$$

$$\otimes (|0\rangle + e^{2\pi i(0.j_1 j_2 j_3)}|1\rangle) \quad (5)$$

這種可拆解性使得 QFT 可以透過一連串單一 qubit 的 Hadamard gate 與數個受控旋轉閘（Controlled $-$ R_k）來高效實現，成為量子相位估計（QPE）等重要演算法的核心模組。

QFT 電路結構與運作流程

QFT的電路結構可分為三個主要部分，分別負責建立疊加、累積相位資訊，以及修正輸出順序。以三個 qubit $|q_2 q_1 q_0\rangle$（等效於上述理論中$|j_1 j_2 j_3\rangle$）為例說明 QFT 電路的運作流程：

1. Hadamard Gate（均勻疊加初始化）

每個 qubit 從最低位 q_0 施加 H gate 操作從右到左遞進，建立基本的疊加形式：

$$=> H|j_m\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_m}|1\rangle)$$

$$,where\ 0.j_m = \begin{cases} 0, if\ \ j_m = 0, \\ 0.5, if\ \ j_m = 1. \end{cases} \quad (6)$$

此操作將 qubit 狀態初始化為均勻疊加，為後續相位干涉鋪路。

2. Controlled - Phase gate

在每個 Hadamard 操作後，會對後方 qubit 依序施加受控相位門 CP gate (Controlled-Phase)，負責將相位資訊依 qubit 間距依序累積。其矩陣定義為：

$$CP(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$$

依照圖 1. 中的電路可知:

- 由 q_1控制 $q_2 \rightarrow q_2$ 接受 $CP(\frac{\pi}{2})$
- 由 q_0控制 $q_2 \rightarrow q_2$ 接受 $CP(\frac{\pi}{4})$
- 由 q_0 控制 $q_1 \rightarrow q_1$ 接受 $CP(\frac{\pi}{2})$

因此可看出 q_2 接受 $CP(\frac{\pi}{2})$與 $CP(\frac{\pi}{4})$，而 q_0 僅接受 H gate。因此各 qubit 的輸出是:

$$|q_0\rangle \xrightarrow{QFT} \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.q_0)}|1\rangle)$$

$$|q_1\rangle \xrightarrow{QFT} \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.q_1 q_0)}|1\rangle)$$

$$|q_2\rangle \xrightarrow{QFT} \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.q_2 q_1 q_0)}|1\rangle) \quad (7)$$

q_2 受到 q_0 與 q_1 控制，也就是理論式(5)的最後一項。其中 $0.q_2 q_1 q_0$ 代表的是一個**二進位小數**：

$$0.q_2 q_1 q_0 = \frac{q_2}{2} + \frac{q_1}{4} + \frac{q_0}{8} \quad (8)$$

這些受控相位旋轉閘模擬出不同階層的相位偏移，透過量子干涉機制，逐步將輸入狀態中的相位資訊「編碼」進 qubit 系統中。每一個 CP gate 對應於一個特定權重的位元貢獻，最終經過 QFT 處理後，各 qubit 所帶的相位干涉資訊能被完整保留下來，為後續的測量與解碼（如在 QPE 中）鋪路。

3. Swap Gate（位元順序修正）

QFT 電路的最後一個步驟是加入 Swap Gate，用來反轉 qubit 的邏輯順

序。因為 QFT 電路是「從最低位元開始處理」，但輸出我們希望呈現成「從最高位元開始顯示」，所以最後必須加入 Swap gate，把 qubit 的邏輯順序反轉回來，對應到正確的二進位表示順序。完整 QFT 如圖 1. 呈現。

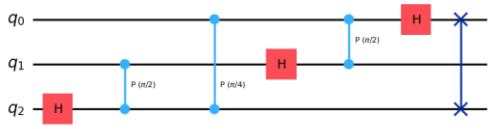

圖 1 . QFT（3 qubit）電路圖

量子相位估計(QPE)

量子相位估計是一種「估測么正運算 U 特徵值中所隱藏相位 ϕ」的演算法，若特徵向量滿足

$$U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle \quad (9)$$

則 QPE 就可以透過一系列量子邏輯操作，有效估計出特徵值中的相位參數 $\phi \in [0,1)$。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，藉由量子干涉與傅立葉轉換（QFT）將其解碼為**二進位表示**。

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：

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 (9)，因此初始整體狀態為：

$$|\psi_0\rangle = |0\rangle^{\otimes n}|\psi\rangle \quad (10)$$

2. 對控制 qubit 套用 H gate

每個控制 qubit 轉為疊加態：

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n}|\psi\rangle \quad (11)$$

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：

$$U^{2^j}|\psi\rangle = e^{2\pi i 2^j\phi}|\psi\rangle \quad (12)$$

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：

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

$$\otimes \dots \otimes (|0\rangle + e^{2\pi i 2^0\phi}|1\rangle) \otimes |\psi\rangle \quad (13)$$

乘積中每一個 qubit 都攜帶一段與 ϕ 相關的相位。這個結構可以進一步整理為：

$$\frac{1}{\sqrt{2^n}} \otimes_{l=1}^{n} [|0\rangle + e^{2\pi i (2^n)(2^{-l})\phi}|1\rangle] \otimes |\psi\rangle \quad (14)$$

儘管控制暫存器在前一階段式(13)是以每個 qubit 單獨的 phase 疊加形式出現，其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(15)。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：

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i\phi k}|k\rangle \otimes |\psi\rangle \quad (15)$$

其中，整數 k 為控制暫存器中各 qubit 的二進位組合，表示為：

$$k = \sum_{l=1}^{n} k_l \cdot 2^{n-l} \quad (16)$$

這裡的 $k_l \in \{0,1\}$ 表示第 l 個 qubit 所處的基底狀態($|0\rangle$或$|1\rangle$)。例如，當 $n = 3$ 且控制暫存器狀態為$|101\rangle$時，即對應十進位整數 $k = 5$。每個基底態$|k\rangle = |k_1 k_2 \dots k_n\rangle$ 皆攜帶一個相位因子：

$$e^{2\pi i\phi k} = e^{2\pi i\phi}\sum_{l=1}^{n} k_l \cdot 2^{n-l} \quad (17)$$

這樣的展開顯示出，原本分散於各個 qubit 的 phase 資訊，最終在控制暫存器中形成一個以 k 為變數的傅立葉型態。這也是後續施加逆量子傅立葉轉換（Inverse QFT）能夠有效解碼出 ϕ 的基礎。

4. 施加 Inverse QFT

回顧 QFT 的作用(式 2):

$$|j\rangle \xrightarrow{QFT} \frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}}|k\rangle$$

接下來，對狀態 $|\psi_2\rangle$ 施加逆量子傅立葉轉換（Inverse QFT），並以式(15)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(18)：

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

$$= \frac{1}{2^n}\sum_{j=0}^{2^n-1}\sum_{k=0}^{2^n-1} e^{-\frac{2\pi ik}{2^n}(j-2^n\phi)}|j\rangle \otimes |\psi\rangle$$

$$= |\psi_3\rangle \quad (18)$$

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 $2^n\phi$。

干涉觀點下的解碼機制：

從干涉的角度來看，若測量結果 j 接近 $2^n\phi$，則各項複數指數 $e^{-\frac{2\pi ik}{2^n}(j-2^n\phi)}$ 的相位近似一致，各項向量將在複數平面上**同方向累加**，產生**建設性干涉**，使得該 $|j\rangle$ 對應的振幅最大。相反地，若 j 與 $2^n\phi$ 差距較大，各項相位將在單位圓上平均分佈，互相抵銷而產生**破壞性干涉**，對應 $|j\rangle$ 的振幅趨近於零。

因此，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。

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：

$$j \approx 2^n\hat{\phi} \quad (19)$$

因此可反推出估計相位為：

$$\hat{\phi} = \frac{j}{2^n} \quad (20)$$

這裡的 j 即為 QPE 最終控制暫存器的量測結果（對應式（18）中傅立葉基底 j）。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：

$$|\phi - \hat{\phi}| < \frac{1}{2^{n+1}} \quad (21)$$

整體電路圖如圖 2.

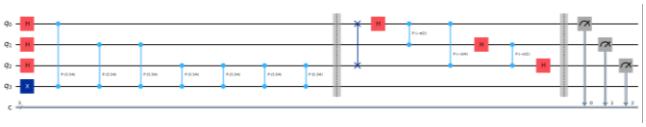

圖 2 . QPE（3 qubit）電路圖

研究及開發目標

本專題旨在探討量子相位估計（Quantum Phase Estimation, QPE）應用於 PSK 信號相位估測的可行性，並實作一套結合軟體模擬與硬體驗證的完整系統。研究與開發目標如下：

1. **建立 QPE 模擬模型：**利用 Qiskit 套件於 Python 環境中建構 QPE 電路，完成受控相位旋轉、逆量子傅立葉轉換與測量流程，驗證演算法正確性。

2. **設計並實現 FPGA 硬體架構**：以 Verilog 撰寫各功能模組，包括除頻器、Moore 狀態機、按鍵控制、七段顯示器與 LED 指示等，搭建完整顯示與輸入控制架構。
3. **支援多種 PSK 模式**：系統具備 BPSK、QPSK 與 8-PSK 切換功能，透過按鈕切換不同模式並顯示相應的相位估計值與誤差。
4. **模擬並測試相位擾動條件**：導入 LFSR 偽隨機模組以產生相位誤差，評估系統在雜訊條件下的穩定性與容錯能力。
5. **整合模擬與實作結果進行驗證**：比對 Qiskit 模擬輸出與 FPGA 顯示結果，評估誤差精度與一致性，驗證系統的準確性與實用性。

參、 專題進行方式

A. 專題開發流程

圖 3. 專題開發流程圖

B. 軟體模擬設計(Qiskit 實作 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 4. 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 5. 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 $|1\rangle$，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 6. 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。

$$\text{Controlled} - \text{phase (CP)} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\theta} \end{bmatrix}$$

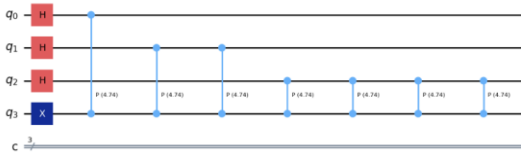

圖 7. Step 1-3 電路圖

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 8. 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 $CP(-\frac{\pi}{2^k})$，這兩步驟就是逆量子傅立葉(IQFT)，把藏在量子狀態中的相位資訊還原出來。

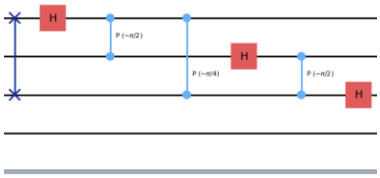

圖 9. IQFT 電路圖

Step5:測量

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

圖 10. Step 5 (Qiskit)

在量測後，原本以疊加與干涉方式儲存的相位資訊，會**塌陷為一組固定的** bit pattern **結果**。

Step6:觀察量子狀態（statevector）

```
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 11. Step 6 (Qiskit)

以 **Bloch 球**視覺化每個 qubit 的量子態

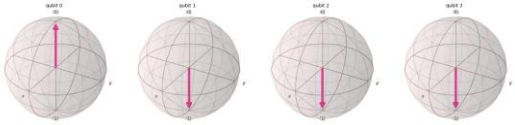

圖 12. Bloch 球顯示測量結果

Step7: 結果解析與相位估計

```
print("Phase estimation output")
a = counts.most_frequent()
print('Most frequent measurement: ',a,'\n')
bin_a = int(a,2)
phase = bin_a/(2**qnum)
print('Actual phase is: ',actual_phase)
print('Estimated phase is: ',phase)
```

圖 13. 測量計算結果

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(20) 的整數近似，因此反推 $\hat{\phi}$ 可用：

$$估計相位\ \hat{\phi} = \frac{int(a,2)}{2^{qnum}} \quad (22)$$

最後兩行將**實際相位值**與**估測結果**一併輸出進行比較。

C. FPGA 實作設計架構

本設計以 Top_BTN 為頂層模組，整合各子模組並負責狀態控制與顯示。整體架構如下：

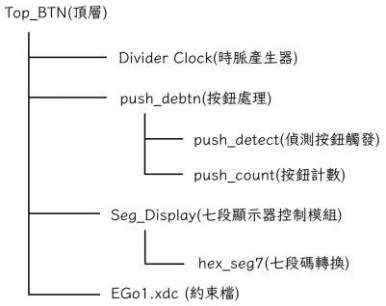

圖 14. FPGA 模組階層圖

---Top_BTN 頂層模組設計邏輯---

狀態跳轉：透過按鍵輸入切換狀態

選擇採用摩爾有限狀態機(Moore Finite State Machine)，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。以下為本系統所設計的狀態轉移條件與對應行為說明：

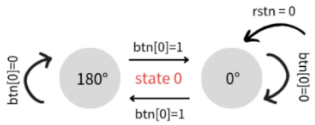

圖 15. BPSK 控制二狀態循環

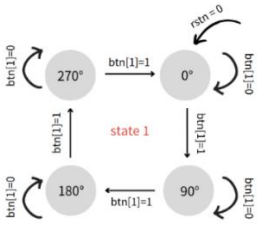

圖 16. QPSK 控制四狀態相位循環

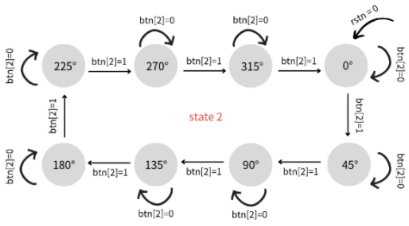

圖 17. 8-PSK 控制八狀態切換

```
always @(posedge clk_100 or negedge rstn) begin
    if (!rstn)
        state2 <= 4'd0;
    else if (btn_r[2]) begin
        case (state2)
            4'd0: state2 <= 4'd1;
            4'd1: state2 <= 4'd2;
            4'd2: state2 <= 4'd3;
            4'd3: state2 <= 4'd4;
            4'd4: state2 <= 4'd5;
            4'd5: state2 <= 4'd6;
            4'd6: state2 <= 4'd7;
            4'd7: state2 <= 4'd8;
            4'd8: state2 <= 4'd0;
            default: state2 <= 4'd0;
        endcase
    end
end
```

圖 18．8-PSK 的狀態機(Verilog)

相位估計邏輯：對應當前狀態執行估算

在本設計中，每個狀態對應一個預設的「相位值」，代表理論角度(如 225°)。為了模擬 QPE 的行為，將實際相位轉為小數格式 $\varphi \in [0,1)$ 再透過 QPE 的推估方式式(20)反推出相位估計值。

→ 相位輸入(以 225° 為例)

$$\text{Phase} = \frac{\text{angle}}{360} = \frac{225}{360} = 0.625 \quad (23)$$

並可額外加入微擾誤差（如+0.001），得 0.626。

→ 數值轉為整數，乘上 10^8 得:

$$\text{raw\_Phase} = \text{Phase} \times 10^8 = 62600000 \quad (24)$$

→ 模擬 QPE 測量（反推公式）

透過整數乘除實作估計相位得：

$$\hat{\phi} = \frac{\text{round}(\text{raw\_phase} \times 256)}{2^{qnum}} \quad (25)$$

此過程對應實作程式片段如下：

```
s5: begin
    led2 = 8'b00011111;
    raw_phase2 = 32'd62600000;   //0.625+0.001
    raw_phase_estimate2 = ((raw_phase2 * 256 + 50000000) / 100000000)* 100000000 / 256;
    end
```

圖 19．模擬 QPE 估計邏輯(Verilog)

其中加上 50000000 為四捨五入補償，而因本設計中 qnum = 8，$2^{qnum} = 256$

此系統雖無法直接模擬量子干涉與測量，但透過數值映射與位移運算，仍可在 FPGA 上近似 QPE 的估計行為，並以視覺化方式驗證估計效果與誤差。

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（pseudo-random sequence）。

當 rstn 為低時，LFSR 重置為 4'b0001；當時脈 clk_100 上升沿到達時，每個時脈週期會將 LFSR **右移一位**，並將最高兩位 lfsr[3] 與 lfsr[2] 做 XOR 運算後的結果插入最低位，如下所示：

```
always @(posedge clk_100 or negedge rstn) begin
if (!rstn)
    lfsr <= 4'b0001;
else
    lfsr <= {lfsr[2:0], lfsr[3] ^ lfsr[2]};
end
```

圖 20．產生偽隨機數列(Verilog)

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 offset_reg（由 lfsr %10×100,000 得到，範圍是 0 到 900,000），再把它加到原始相位 raw_phase 上，模擬測量誤差或雜訊干擾。

```
05      always @(posedge clk_100 or negedge rstn) begin
06      if (!rstn)
07          offset_reg <= 0;
08      else if (btn_r[0] | btn_r[1] | btn_r[2])  // 三
09          offset_reg <= (lfsr % 10) * 32'd100000;
10      end
```

圖 21．將偽隨機數列倍增至 offset(Verilog)

表一：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
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

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。

--- 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。

假設輸入 clkin 為 50MHz，若每計數 50,000 個時脈反轉一次輸出時脈，則：

$$\frac{50 \times 10^6}{2 \times 50000} = 500Hz \quad (26)$$

因此若要產生 1kHz 輸出，需每計數 25,000 個時脈反轉一次，即一完整週期為 50,000 個時脈。

```
always@(posedge clkin or negedge rst_n)begin
    if(!rst_n) begin
        Counter_1k <= 0;
        Counter_100 <= 0;
        Counter_10 <= 0;
        Counter_1 <= 0;
    end else begin
        //1KHz
        if(Counter_1k == (Divider_Counter_1K - 1))
            Counter_1k <= 0;
        else
            Counter_1k = Counter_1k + 1;
        //100Hz
        if(Counter_100 == (Divider_Counter_100 - 1))
            Counter_100 <= 0;
        else
            Counter_100 <= Counter_100 + 1;
        //10Hz
        if(Counter_10 == (Divider_Counter_10 - 1))
            Counter_10 <= 0;
        else
            Counter_10 <= Counter_10 + 1;
        //1Hz
        if(Counter_1 == (Divider_Counter_1 - 1))
            Counter_1 <= 0;
        else
            Counter_1 <= Counter_1 + 1;
    end
end
```

圖 22 . 除頻器計算時脈主要概念(Verilog)

表二：各 module 使用到的頻率表

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。

```
    //H2L detect btn_de @ 100hz for each button
    assign btn_r[0] = btn_de[0] & (!btn_del[0]);
    assign btn_r[1] = btn_de[1] & (!btn_del[1]);
    assign btn_r[2] = btn_de[2] & (!btn_del[2]);
    assign btn_r[3] = btn_de[3] & (!btn_del[3]);
    assign btn_r[4] = btn_de[4] & (!btn_del[4]);

    //delay the btn_de with 100hz for 1 clock cycle
    always@(posedge clk_100 or negedge rstn)
    begin
        if(!rstn)
            begin
                btn_del <= 0;
            end
        else
            begin
                btn_del <= btn_de;
            end
    end
endmodule
```

圖 23 . 去按鈕雜訊(Verilog)

--- 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。

```
        always @( posedge clk or negedge rstn ) //1
            if( !rstn )
                begin
                    state <= 5'd0;
                    pos <= 5'd0;
                end
            else
                case( pos )
                    5'd0:
                        begin
                            state <= 5'd0;
                            pos <= btn;
                        end
                    5'd1, 5'd2, 5'd4, 5'd8, 5'd16:
                        begin
                            state <= pos;
                            pos <= 5'd0;
                        end
                    default:pos <= 5'd0;
                endcase
endmodule
```

圖 24 . 偵測按鈕觸發（Verilog)

只有當 pos == 5'd0 時才會讀進新的按鈕狀態 btn，然後 pos 會變成一個

One-Hot 值，如此可以確保每次按鍵只觸發一次狀態變更，而且跳轉很乾淨。

--- 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。

```
always @( posedge clk or negedge rstn )
begin
    if( !rstn )
        begin
            pos <= 5'd0;
            btn_de <= 5'd0;
        end
    else
        case( pos )
            5'd0:
                begin
                    btn_de <= 5'd0;
                    pos <= state;
                end
            5'd1, 5'd2, 5'd4, 5'd8, 5'd16:
                if( !cnt_sig )
                    pos <= 5'd0;
                else
                    btn_de <= pos;
            default:
                pos <= 5'd0;
        endcase
end

endmodule
```

圖 25 . 按鈕計數(Verilog)

可確保**只有在持續按壓狀態時，才更新 btn_de 為目前狀態**。一旦 cnt_sig 變為 0（已放開），則 pos 被清除，回到初始等待狀態。

--- 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。

```
always@(posedge Scan_clk or negedge rtsn)begin
    if(!rtsn) begin
        Seg7_show <= 8'b11111111;
        state <= 0;
        number <= 5'b00000;
    end else begin
        if(state < 8)
            state <= state + 1;
        else
            state = 0;
        case(state)
            4'd0:
                begin
                    Seg7_show <= 8'b1000_0000;
                    number <= {0,Number0};
                end
            4'd1:
                begin
                    Seg7_show <= 8'b0100_0000;
                    number <= {0,Number1};
                end
            4'd2:
                begin
                    Seg7_show <= 8'b0010_0000;
                    number <= {0,Number2};
                end
            4'd3:
                begin
                    Seg7_show <= 8'b0001_0000;
                    number <= {0,Number3};
                end
            4'd4:
                begin
                    Seg7_show <= 8'b0000_1000;
                    number <= {0,Number4};
                end
            4'd5:
                begin
```

圖 26 . 七段顯示器(Verilog)

----- 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。

```
module hex_seg7 (hex,seg_data);
  input [4:0] hex;
  output reg [7:0] seg_data;

    always@ (hex)begin
        case (hex)
            5'b0_0000 : seg_data  = ~(8'b1100_0000);//0
            5'b0_0001 : seg_data  = ~(8'b1111_1001);//1
            5'b0_0010 : seg_data  = ~(8'b1010_0100);//2
            5'b0_0011 : seg_data  = ~(8'b1011_0000);//3
            5'b0_0100 : seg_data  = ~(8'b1001_1001);//4
            5'b0_0101 : seg_data  = ~(8'b1001_0010);//5
            5'b0_0110 : seg_data  = ~(8'b1000_0010);//6
            5'b0_0111 : seg_data  = ~(8'b1111_1000);//7
            5'b0_1000 : seg_data  = ~(8'b1000_0000);//8
            5'b0_1001 : seg_data  = ~(8'b1001_0000);//9
            5'b1_0000 : seg_data  = ~(8'b0100_0000);//0.
            default : seg_data  = ~(8'b0100_0000); //顯示 0.
        endcase
    end
endmodule
```

圖 27 . 七段顯示轉碼對應格式(Verilog)

肆、 主要成果與評估

- **Qiskit 模擬結果與理論驗證**

量子電路圖：

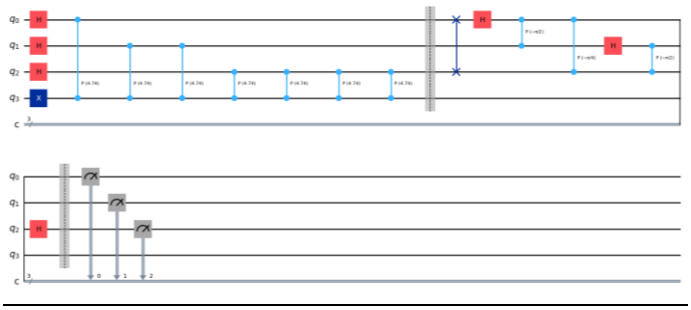

圖 28 . QPE on 3 qubit 電路圖

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：

$$angle = 2\pi \times actual\ phase \quad (27)$$
$$= 2\pi \times 0.754 = 4.7375$$

Bloch 球圖觀察 qubit 狀態

Bloch 球圖呈現的是**量子期望方向**，q0,q1,q2（計數 qubits）中：q0 指向 | 0 ⟩（Bloch 球朝上）q1, q2 指向| 1 ⟩（Bloch 球朝下）對應二進位表示為

$$q_2 q_1 q_0 = 110$$

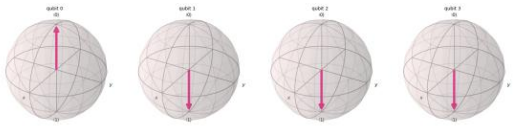

圖 29．Bloch 球圖觀測 QPE on 3 qubit

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下圖。

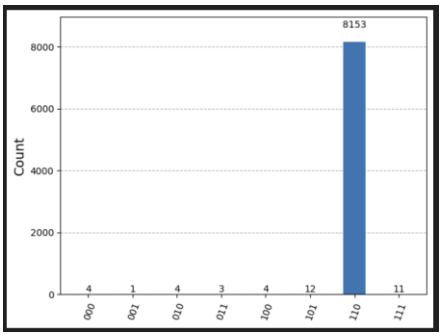

圖 30．量測結果分布

可看出**最大值為 110**，出現 **8153 次**，佔總次數的 99.5%。

數據結果輸出：

在本次實驗中，最常見輸出為 110，使用 3 個 qubit 對應的相位估計值是：

$$\hat{\phi} = 0.110_2 = \frac{6}{2^3} = 0.75 \quad (28)$$

```
Phase estimation output
Most frequent measurement:  110

Actual phase is:  0.754
Estimated phase is:  0.75
```

圖 31．量測結果數據顯示

而原本設定的真實相位 = 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，我們進一步針對整個相位範圍進行模擬。具體而言，我們將 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。

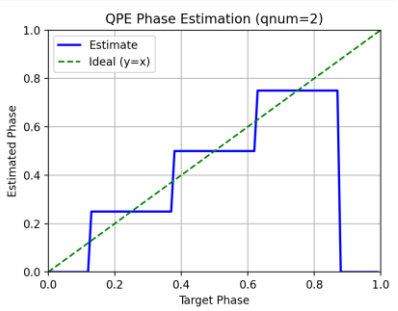

圖 32．qnum = 2 之 QPE 模擬與理論比對結果

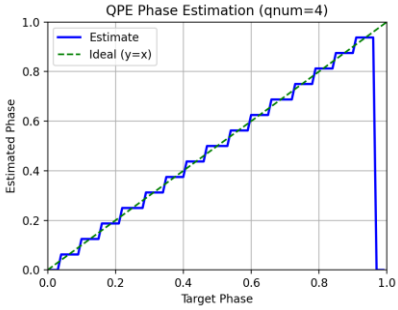

圖 33．qnum = 4 之 QPE 模擬與理論比對結果

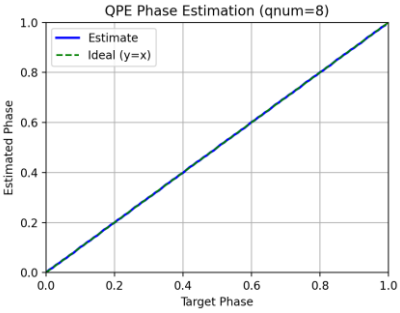

圖 34. qnum = 8 之 QPE 模擬與理論比對結果

圖 32-34 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。

- **FPGA 顯示輸出成果**

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

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。

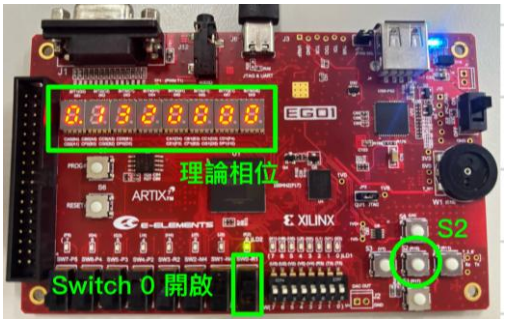

圖 35. 8-PSK 第一跳轉理論相位

圖 35.顯示理論值為 0.125 加上 0.007 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。

圖 36. 8-PSK 第一狀態估計相位

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。

```
Phase estimation output
Most frequent measurement:  00100010

Actual phase is:  0.132
Estimated phase is:  0.1328125
```

圖 37. 使用 Qiskit 執行結果

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。

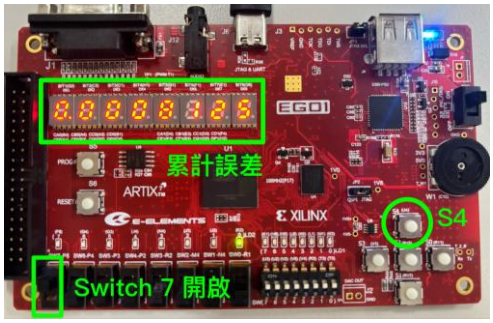

圖 38. 8-PSK 第一狀態理論&估計誤差

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。

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 39. 8-PSK 第二跳轉理論相位

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）。

4. 顯示估計值與累積平均誤差

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。

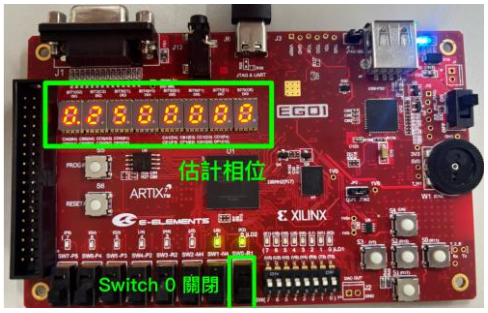

圖 40. 8-PSK 第二狀態估計相位

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。

```
Phase estimation output
Most frequent measurement:  01000000

Actual phase is:  0.251
Estimated phase is:  0.25
```

圖 41．使用 Qiskit 執行結果

圖 42．8-PSK 第二狀態理論&估計誤差

計算方式如下，並取至小數點後七位：

$$AvgError = \frac{1}{N}\sum_{i=1}^{N}|\hat{\phi}_i - \phi_i| \quad (29)$$

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：

$$\frac{1}{2}(0.0008125 + 0.001) = 0.00090625 \quad (29)$$

- **紀錄模擬與實測誤差比較**

表二：模擬與實測結果數據紀錄表

理論相位	Qiskit	FPGA
0.0010000	0.0	0.0000000
0.0060000	0.0078125	0.0078125
0.0070000	0.0078125	0.0078125
0.1310000	0.1328125	0.1328125
0.2510000	0.25	0.2500000
0.2570000	0.2578125	0.2578125
0.3810000	0.3828125	0.3828125
0.3820000	0.3828125	0.3828125
0.5060000	0.5078125	0.5078125
0.5070000	0.5078125	0.5078125
0.6260000	0.625	0.6250000
0.6310000	0.6328125	0.6328125
0.6320000	0.6328125	0.6328125
0.7510000	0.75	0.7500000
0.7520000	0.75390625	0.7539062
0.7560000	0.7578125	0.7578125
0.7570000	0.7578125	0.7578125
0.8760000	0.875	0.8750000
0.8820000	0.8828125	0.8828125
平均誤差	0.00127	0.00127
最大誤差	0.00191	0.00191

根據模擬與實測結果的數據分析，除了顯示出的小數位不同以外，整體顯示出本設計中所實作之 FPGA 數位估測邏輯，已成功模擬 QPE 的預期行為，能夠有效地在硬體中實現量子相位估計過程，並重現模擬電路所產生的估測結果。

- **預期及實際成效差距**

1. 語言不熟、摸索時間長

由於初期對 FPGA 與 **Verilog** 語言較為**不熟悉**，導致在模組撰寫與除錯階段耗費大量時間。特別是在理解**邊緣觸發、模組間訊號傳遞**等基礎概念時，需花費額外時間查閱資料與實作驗證。

2. 輸入方式簡化

原先預期可利用旋轉式滾輪(Rotary Encoder)搭配 ADC(類比數位轉換器）讀取類比電壓轉為相位輸入，模擬更細緻的連續相位調整，但由於系統設計與整合難度較高，因此最終改採用按鈕作為相位切換機制。

3. 顯示方式調整

原計畫預想是在程式撰寫時，直接使用**輸出浮點數相位**，但在硬體架構中實作**浮點數乘除與顯示過度繁瑣**，因此改採用定點策略處理，及**固定顯示格式開頭為 0.**，後方七位數經乘上

10^8 處理後再轉為七段輸出，達成兼顧準確性與硬體簡化的效果。

4. 誤差計算方式簡化

最初預期導入 RMSE(均方根誤差)進行估計誤差分析，但因平方開根號運算在 FPGA 上需使用 DSP slice 或 IP 核，占用資源較大，最終改以平均絕對誤差，近似評估準確度，仍能有效反映估計與實際間的偏差程度。

5. 資源受限導致量子邏輯電路無法完整還原

QPE 原理中包含多組受控旋轉閘與反量子傅立葉轉換(Inverse QFT)結構，若欲完整實現需仰賴真正的量子運算平台。由於 FPGA 屬於經典硬體架構，無法實現真正的量子閘運算，但可透過邏輯設計模擬量子演算法的行為與預期輸出結果。

伍、 結語與展望

本專題整合 Qiskit 模擬與 FPGA 實作，成功建立一套以量子相位估計(QPE)為核心的相位估測系統。軟體部分以 Qiskit 架構完整量子電路，驗證估測邏輯的正確性；硬體部分則以 Verilog 撰寫 Top_BTN 模組，整合除頻器、狀態機、顯示器與偽隨機擾動，將估測邏輯移植至 Artix-7 FPGA 並完成系統整合。

本設計雖尚未對傳統通訊系統帶來直接實用效益，但作為一種概念驗證模型，展示了量子相位估計（QPE）可作為接收端邏輯，用以辨識經典調變相位（如 BPSK、8-PSK）並估算其相對誤差。主要強調跨域整合的潛力，並可作為未來研究「量子化訊號解調器」或量子感測應用的基礎。

未來若有機會，期望能**結合實際量子晶片（如 IBM Quantum）**進行跨平台驗證與混合模擬，提升系統的應用深度與廣度。同時亦可進一步導入誤差更正機制（Quantum Error Correction），探討其在實際通訊與運算場景中的可行性。本專題在理論模擬與硬體實作之間搭起橋樑，提供一種以數位邏輯近似量子估測行為的設計方法，未來可延伸應用於量子模擬、智慧通訊與異質整合系統的開發中。

陸、 後記及銘謝

本專題歷經模擬驗證、硬體實作與系統整合等多階段開發，過程中不僅深化了我對量子計算與數位邏輯設計的理解，也累積了寶貴的實作經驗。在此，謹向指導老師 沈瑞欽教授致上最誠摯的感謝，感謝老師於專題期間給予我悉心指導與技術上的建議，使本專題得以順利完成。

柒、 參考文獻

[1] 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.

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

[3] B. Sklar, *Digital Communications: Fundamentals and Applications*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2001.

[4] J. H. Shapiro and S. Lloyd, “Quantum illumination versus coherent-state target detection,” *New J. Phys.*, vol. 9, no. 8, p.

200, Aug. 2007.
[5] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2010.
[6] 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
[7] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2010.
[8] IBM Quantum, "PhaseEstimation class — Qiskit documentation," *Qiskit API Reference*, [Online]. Available on, https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.PhaseEstimation