EOS 期末報告

系級: 電機四

組員: 陳冠維、林庭毅、陳祥均、羅豐祥

## 1 AES(解密未實現)

暫存器分配(14 個暫存器、32-bits data width)

### **1.1 系統控制信號**

* **clk** : 系統時鐘（設計頻率：200MHz，週期5ns）
* **reset** : 重置信號（高電平有效，同步重置）
* **start** : 啟動加密/解密操作（正邊緣脈衝觸發）
* **done** : 操作完成信號（高電平表示完成）

### **1.2 模式控制**

* **mode** : 模式選擇信號
  + 1 = 加密模式（Encryption）
  + 0 = 解密模式（Decryption）

### **1.3 資料載入控制**

* **load** : 載入控制信號
  + 需要**兩個時鐘週期**完成128位元資料載入
  + 第一週期：load 0→1 觸發高64位元載入
  + 第二週期：load 1→0 觸發低64位元載入

### **1.4 資料介面**

* **key[63:0]** : 64位元密鑰輸入
  + 總共需載入兩次完成128位元密鑰
  + 載入順序：先高位[127:64]，後低位[63:0]
* **data\_in[63:0]** : 64位元明文/密文輸入
  + 總共需載入兩次完成128位元資料
  + 載入順序：先高位[127:64]，後低位[63:0]
* **data\_out[127:0]** : 128位元輸出資料
  + 一次性輸出完整的128位元結果

### **1.5 載入時序示例**

週期1: load=1, key[63:0]=key[127:64], data\_in[63:0]=plaintext[127:64]

週期2: load=0, key[63:0]=key[63:0], data\_in[63:0]=plaintext[63:0]

週期3+: load=0, start=1 (開始運算)

### **1.6 加密&解密過程（10回合，以加密為例）**

1. **初始回合密鑰加法** (Round 0)
   * AddRoundKey：明文⊕初始密鑰
2. **主要回合** (Round 1~9)
   * **SubBytes**: S-box位元組替換
   * **ShiftRows**: 行位移轉換
   * **MixColumns**: 混合行運算
   * **AddRoundKey**: 回合密鑰加法
3. **最終回合** (Round 10)
   * **SubBytes**: S-box位元組替換
   * **ShiftRows**: 行位移轉換
   * **AddRoundKey**: 最終回合密鑰加法（跳過MixColumns）

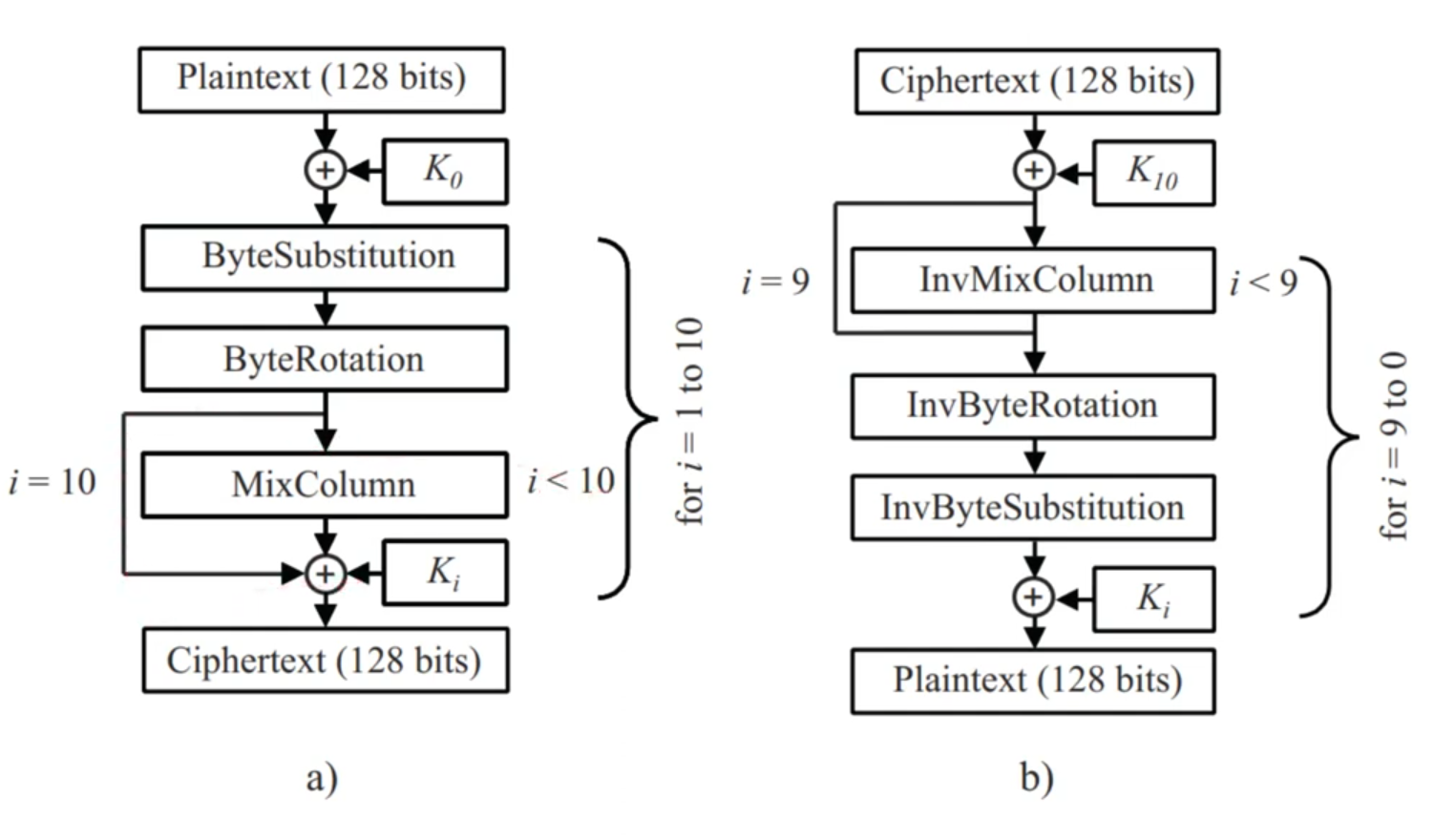


圖1.1.1 加密與解密流程圖

### **1.7 解密失敗原因分析**

**最關鍵原因--load存在控制競爭**

1. **數據載入和密鑰擴展的時序不同步：**當我在輸入時序加入peridoc\_clock時，每次解密的輸出都不同。
2. **解密時的密鑰調度順序可能有誤：**

加密時：

* 數據在輪 1-9 需要 MixColumns ✓（代碼正確）
* 數據在輪 10 不需要 MixColumns ✓（代碼正確）
* 密鑰永遠不需要 MixColumns ✓（代碼正確）

### 解密時（等效解密算法）：

* 數據在解密輪 1 不需要 InvMixColumns ✓（代碼正確）
* 數據在解密輪 2-10 需要 InvMixColumns ✓（代碼正確）
* 密鑰在使用輪 1-9 的密鑰時需要 InvMixColumns
  + 但代碼在 round\_cnt = 1（使用輪 9 密鑰）時沒有處理 ❌
  + 代碼在 round\_cnt = 10（使用輪 0 密鑰）時正確地沒有處理 ✓

1. **MixColumns 在解密流程中的應用時機不正確**

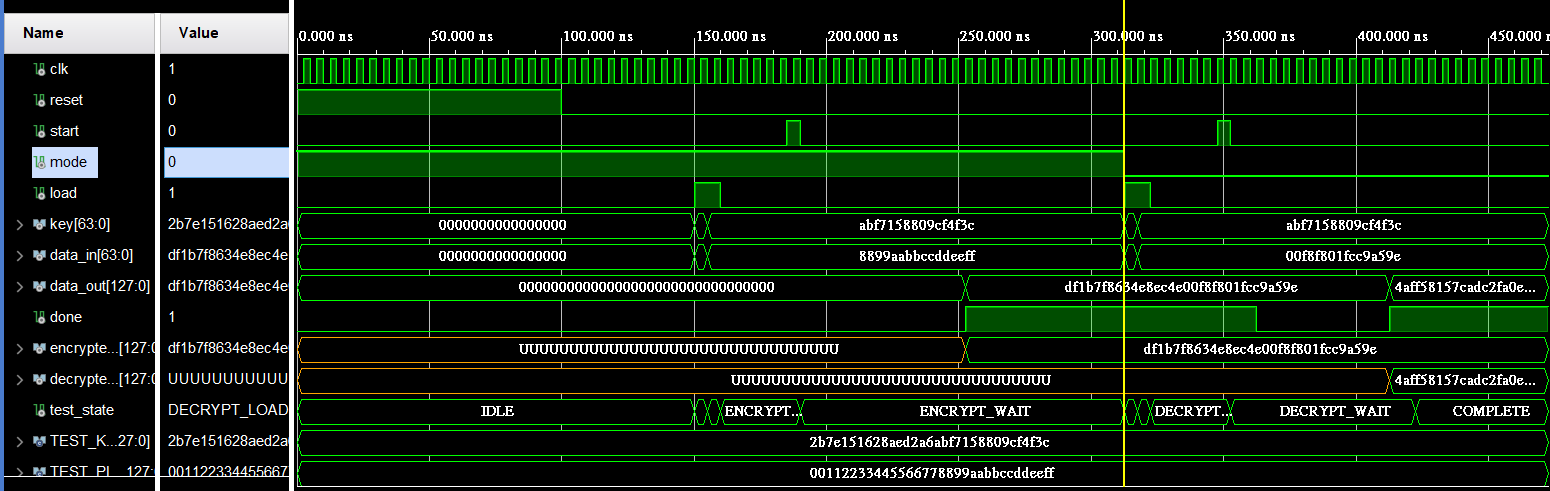


圖1.1.2 testbench時序圖(黃線以左(mode 1 加密)，黃線以右(mode 1 解密))

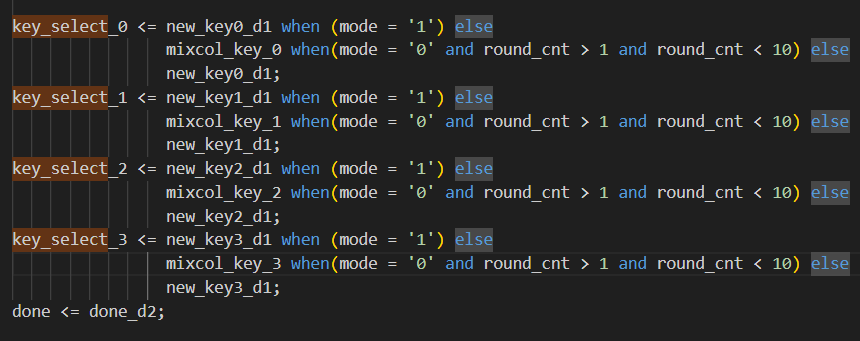


圖1.1.2 錯誤模塊

## 

## **2 DES**

### **2.1 暫存器配置**

* **實際暫存器數量**: 16個暫存器 (slv\_reg0 到 slv\_reg15)
* **資料寬度**: 32位元 (不是128位元)
* **時鐘**: S\_AXI\_ACLK (AXI系統時鐘)
* **重置**: S\_AXI\_ARESETN (低電平有效，異步重置)

### **2.2. 加密演算法**

* **實際演算法**: **DES** (Data Encryption Standard)
* **資料寬度**: 64位元 (不是128位元AES)
* **密鑰長度**: 64位元 (不是128位元)

### **2.3. 暫存器分配**

| **功能** | **64-bit 寄存器對應** | **低位 (bits [31:0])** | **高位 (bits [63:32])** |
| --- | --- | --- | --- |
| **金鑰 (Key)** | key\_std = {slv\_reg3, slv\_reg2} | slv\_reg2 | slv\_reg3 |
| **輸入資料 (Plaintext／Ciphertext)** | data\_bus\_std = {slv\_reg1, slv\_reg0} | slv\_reg0 | slv\_reg1 |
| **輸出結果 (Result)** | 寄存器寫回：slv\_reg5 + slv\_reg6 | slv\_reg5 (只讀) | slv\_reg6 (只讀) |

### **2.4. 操作流程**

1. 寫入64位元輸入資料到 slv\_reg0 (低32位) 和 slv\_reg1 (高32位)
2. 寫入64位元密鑰到 slv\_reg2 (低32位) 和 slv\_reg3 (高32位)
3. 設定 slv\_reg4[1] 選擇加密/解密模式
4. 設定 slv\_reg4[0] = 1 啟動操作
5. 等待 slv\_reg7[0] = 1 表示操作完成
6. 從 slv\_reg5 和 slv\_reg6 讀取64位元輸出結果

### **2.5. DES演算法特點**

* **回合數**: 16回合 (AES是10回合)
* **區塊大小**: 64位元
* **密鑰長度**: 64位元 (實際有效的只有56位元)

### **2.6. 位元順序轉換**

代碼中有大量的位元順序轉換，將標準的 [63:0] 格式轉換為DES模組使用的 [1:64] 格式，其中：

* [1:64] 格式：位元1為LSB，位元64為MSB
* [63:0] 格式：位元0為LSB，位元63為MSB

### **2.7. 測試**

這段測試程式碼的邏輯是針對一個加密模組進行驗證的，主要透過 AXI 寄存器介面與硬體模組進行溝通。首先寫入明文資料到對應的暫存器（slv\_reg0），再寫入金鑰到另一組暫存器（slv\_reg2 和 slv\_reg3）。接著讀取這些值以確認資料已正確寫入。之後，透過寫入控制寄存器來觸發加密操作（bit0=1 表示啟動加密）。啟動後進入輪詢機制，不斷檢查狀態寄存器的 bit0 是否變為 1，表示加密完成；若逾時則跳出。完成後讀取結果寄存器（slv\_reg5 與 slv\_reg6）以取得加密後的密文，並將低位與高位組合成一個 64-bit 的結果。整體邏輯是「設定→啟動→等待→讀取結果」，用來驗證加密硬體模組的正確性與功能。

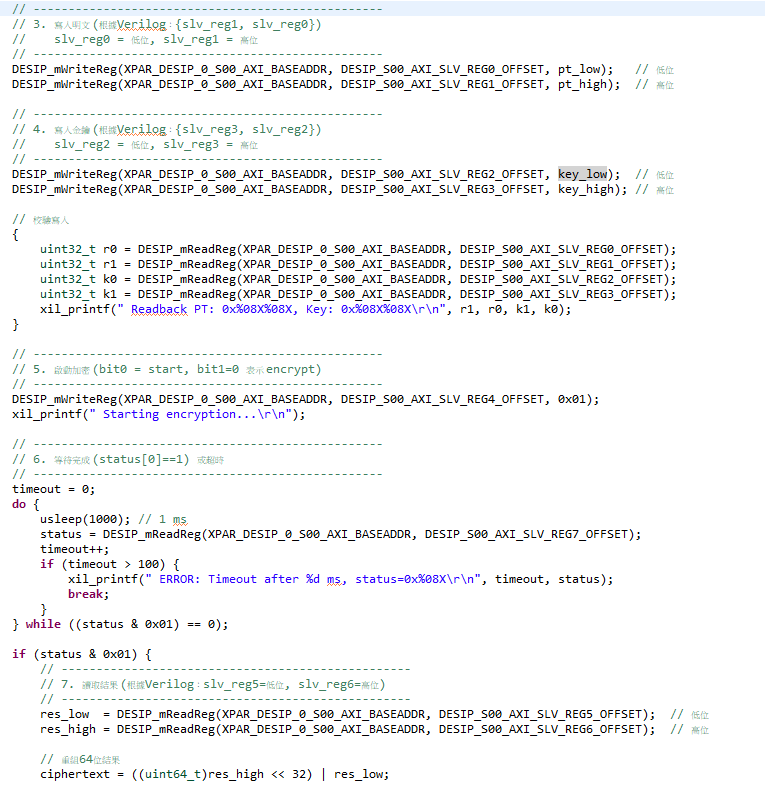


圖2.7.1 測試代碼

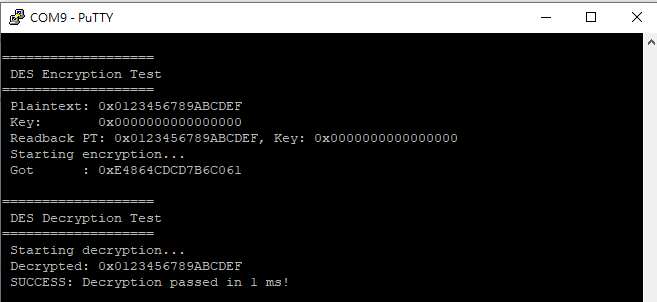


圖2.7.2 測試結果輸出

## **3. GCD**

### **3.1 暫存器配置**

* **實際暫存器數量**: 4個暫存器 (slv\_reg0 到 slv\_reg3)
* **資料寬度**: 32位元
* **時鐘**: S\_AXI\_ACLK (AXI系統時鐘)
* **重置**: S\_AXI\_ARESETN (低電平有效，異步重置)

### **3.2. 計算演算法**

* **實際演算法**: **GCD** (Greatest Common Divisor - 最大公因數)
* **輸入資料寬度**: 8位元 (使用32位元暫存器的低8位)
* **輸出資料寬度**: 8位元
* **計算方法**: 歐幾里得演算法

### **3.3. 暫存器分配**

|  |  |  |  |
| --- | --- | --- | --- |
| **功能** | **暫存器** | **位元範圍** | **說明** |
| **輸入數值 X** | slv\_reg0 | [7:0] | 第一個輸入數值 (僅使用低8位) |
| **輸入數值 Y** | slv\_reg1 | [7:0] | 第二個輸入數值 (僅使用低8位) |
| **控制暫存器** | slv\_reg2 | [0] | 啟動信號 (寫入1啟動計算) |
| **輸出結果** | slv\_reg3 | [7:0] | GCD計算結果 **(只讀)** |

### 

### **3.4. 操作流程**

1. 寫入第一個8位元數值到 slv\_reg0[7:0]
2. 寫入第二個8位元數值到 slv\_reg1[7:0]
3. 設定 slv\_reg2[0] = 1 啟動GCD計算
4. 系統自動檢測啟動脈衝並開始計算
5. 等待計算完成（通常在1ms內完成）
6. 從 slv\_reg3[7:0] 讀取8位元GCD結果

### **3.5. GCD演算法特點**

* **演算法類型**: 歐幾里得演算法 (Euclidean Algorithm)
* **計算複雜度**: O(log(min(x,y)))
* **硬體實現**: 迭代式設計，節省硬體資源
* **計算時間**: 根據測試結果，所有計算皆在2ms內完成，但仍需增加狀態否則結果只會取回0

### **3.6. 脈衝檢測機制**

代碼中實現了完整的脈衝檢測系統，我透過兩變數 start\_pulse, done\_pulse 去檢測開始與結束計算的過程，確保計算的正確啟動和結束。

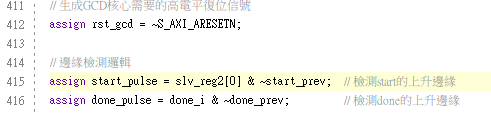


圖3.6.1 脈衝檢測機制代碼

### **3.7. 狀態管理**

透過更改gcdip.vhd (line 141)，將傳出訊號done (脈衝波)，作為計算完成的狀態訊號，並在實例化gcd計算加入done端口得到狀態 ( gcdip\_v1\_0\_S00\_AXI.v line 460)

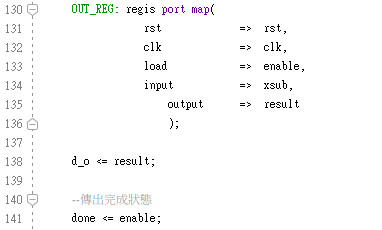


圖3.7.1狀態輸出代碼結構

* **start\_prev**: 儲存前一個時鐘週期的啟動狀態，用於邊緣檢測
* **done\_prev**: 儲存前一個時鐘週期的完成狀態，用於邊緣檢測
* **reg\_start\_i**: 內部啟動信號，在檢測到啟動脈衝時設為高電平，完成時清除

### **3.8. 測試結果範例**

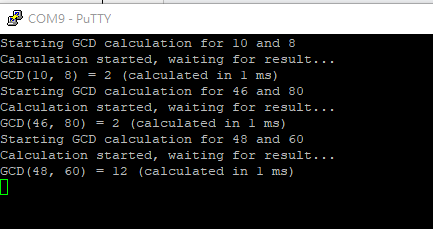


圖3.8.1 gcd測試結果輸出

## **4. 電路整合**

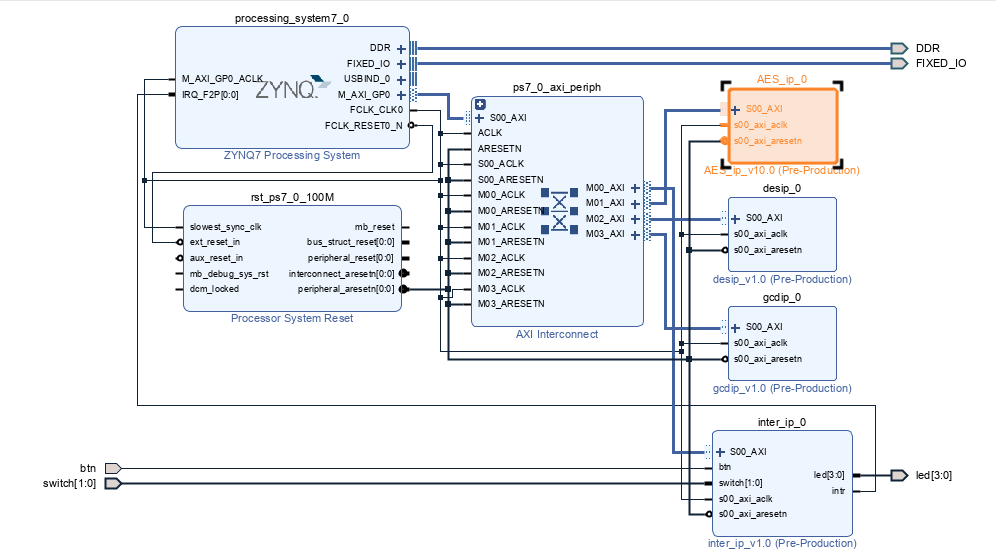


圖4.1.1 完整電路輸出

#### **4.1 處理器系統 與 AXI 匯流排連接**

使用 ZYNQ7 Processing System 做為主控核心，透過 M\_AXI\_GP0 匯流排與 AXI Interconnect 相連，負責與各個自訂 IP 模組進行 AXI4-Lite 的通訊。並透過 ps7\_0\_axi\_periph（AXI Interconnect），將 PS 與三個 IP 使用 AXI-Lite Slave 通道（S00\_AXI）進行對接。包含AES\_ip\_0（加密模組）、desip\_0（可能為 DES 加密模組）、gcdip\_0（最大公因數運算模組）：

#### **4.2 Reset 與時脈控制**

* 利用 Processor System Reset 模組產生各模組所需的 reset 訊號（s00\_axi\_aresetn）與時脈（s00\_axi\_aclk），確保所有模組同步啟動。
* 系統時脈由 Zynq PS 提供，100MHz 或其他頻率來源。

#### **4.3 中斷整合**

* 你將中斷來源 inter\_ip\_0（整合了按鍵與開關的輸入模組）連接到 PS 的 IRQ（IRQ\_F2P[0:0]），讓 ARM 核心能接收來自 PL 端的中斷事件。
* 此模組同時處理 btn、switch 輸入與 led 控制，形成一個用於除錯與互動的界面。

## **5. Standalone**

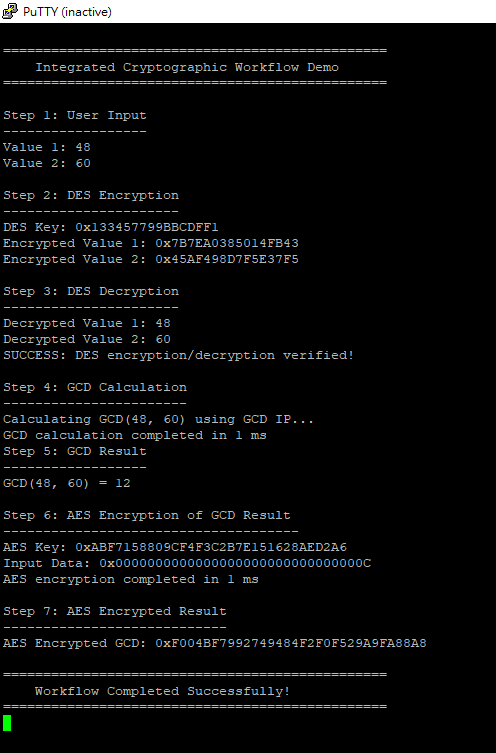
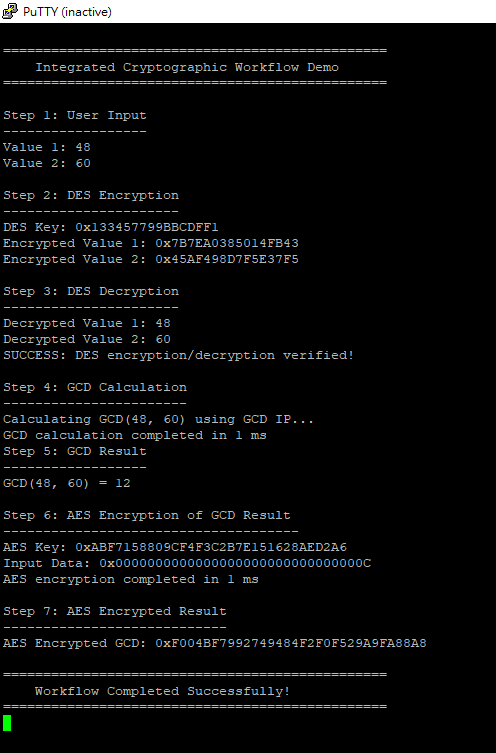
 

圖5.1.1 Standalone結果輸出

整個加密流程如下：

* **DES加解密**：輸入明文和產生的密文對照，顯示加密有效。
* **GCD計算結果**：計算完成的時間與GCD結果（12）。
* **AES加密GCD結果**：明確展示AES加密輸入值與輸出的加密後密文。

### **5.1 流程步驟：**

1. **使用者輸入**：使用者提供兩個整數（例如48和60）。
2. **DES加密**：透過DES硬體IP，以64位元金鑰（如0x133457799BBCDFF1）將輸入值進行加密。
3. **DES解密**：使用DES硬體IP解密取出的資料，並以解密數據進行後續計算。
4. **GCD計算**：計算最大公因數（GCD），如輸入值48與60，則計算結果為12。
5. **AES加密GCD結果**：將GCD計算結果透過AES硬體IP（128位元金鑰，如0xABF7158809CF4F3C2B7E151628AED2A6）進行加密。
6. **結果輸出**：系統完成AES加密後輸出最終密文，如0xF004BBF7992749484F2F0F529A9FA8A8。

### **5.2 關鍵程式碼片段**



圖5.2.1 Standalone關鍵程式碼片段

## **6.FreeRTOS**

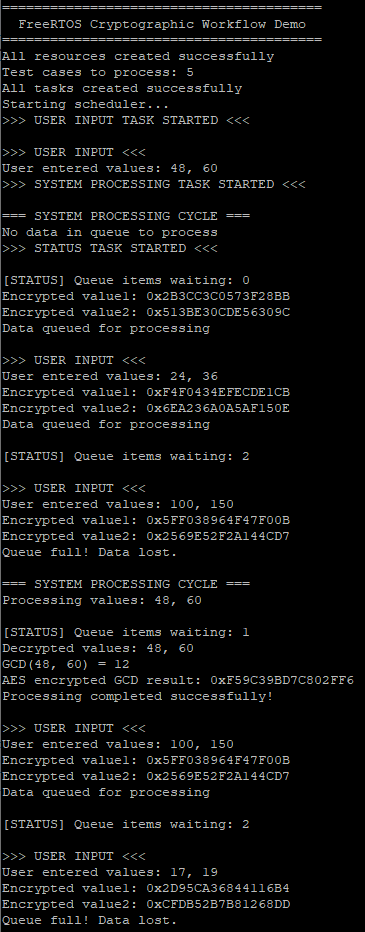
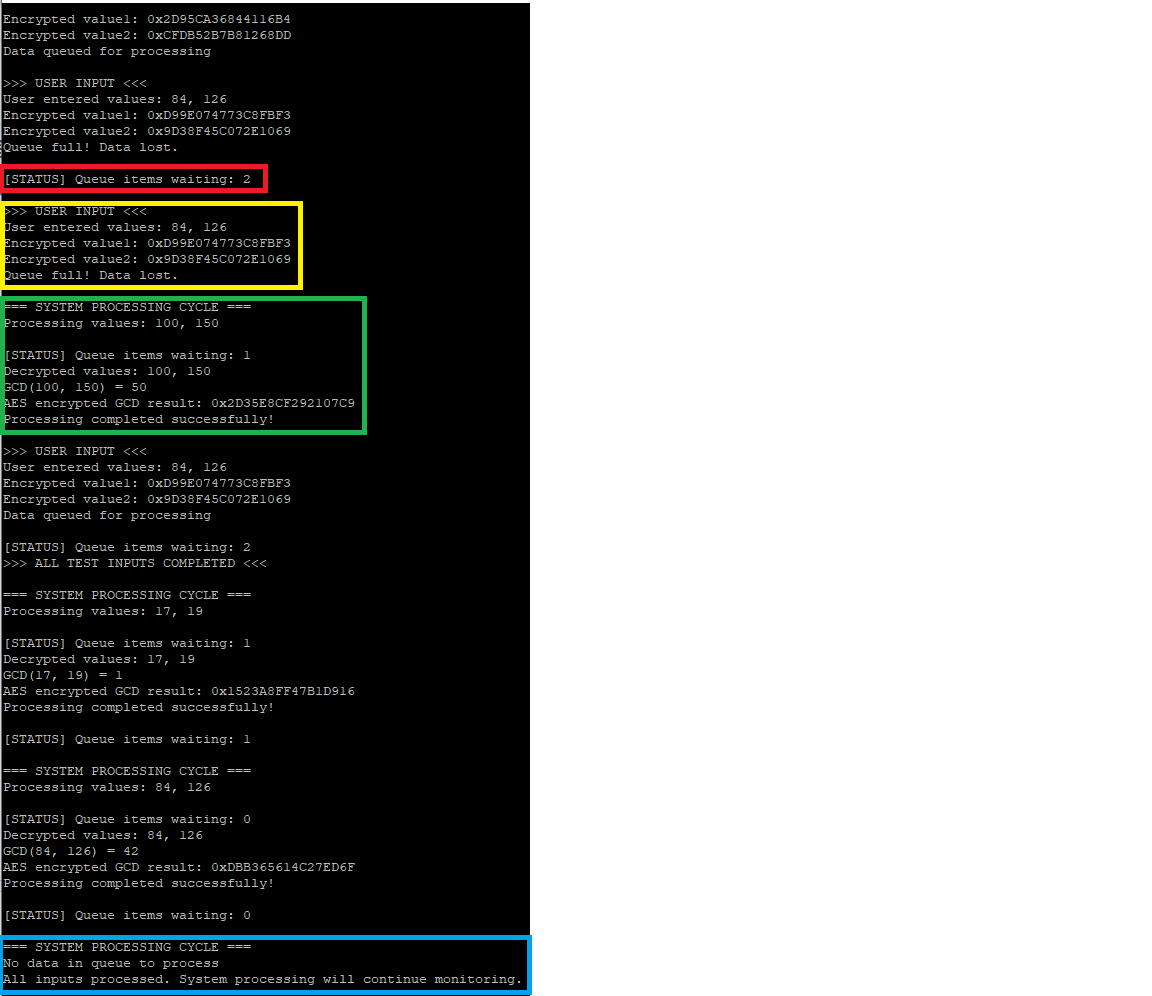
 

圖6.1.1 FreeRTOS結果輸出

上圖結果說明

|  |  |  |
| --- | --- | --- |
| **顏色** | **說明** | **詳細** |
| 紅色 | 佇列等待的元素數量 | 即時顯示 uxQueueMessagesWaiting(xInputQueue) |
| 黃色 | 使用者輸入階段 | 顯示 vUserInputTask 讀取並加密的原始值及密文 |
| 綠色 | 系統計算階段 | 顯示 vSystemProcessTask 解密後的原始值、GCD 計算過程及 AES 加密結果 |
| 藍色 | 處理完成後資訊 | 所有測試數據完成後佇列為空，顯示系統進入監控模式 |

**6.1 系統流程概覽 與 同步機制**

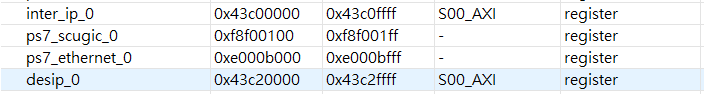
* **使用者輸入任務 (優先權 2):** 每 4 秒讀取一組測試數據，透過 DES IP 核心加密後，使用 `xQueueSend()` 放入佇列。
* **系統處理任務 (優先權 1):** 每 10 秒呼叫 `xQueueReceive()` 取出一筆加密數據。使用 DES IP 核心解密，計算 GCD，再透過 AES IP 核心對 GCD 結果加密，最後輸出。
* **狀態監控任務 (優先權 1):** 每 5 秒顯示佇列當前元素數量，使用 `xSemaphoreTake(xPrintMutex)` 保護印出操作。
* **佇列：**`xInputQueue = xQueueCreate(2, sizeof(UserInput\_t));` 容量 2。

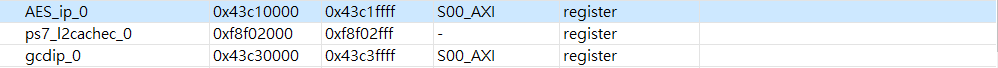
**6.2 關鍵程式片段**



圖6.2.1FreeRTOS關鍵程式

## **7.Linux Driver**

7.1 ip Address  




7.2 ip table

|  |  |  |  |
| --- | --- | --- | --- |
| **IP名稱** | **基地址** | **地址範圍** | **Device Tree節點** |
| inter\_ip\_0 | 0x43c00000 | 0x43c00000-0x43c00fff | inter\_ip@43c00000 |
| AES\_ip\_0 | 0x43c10000 | 0x43c10000-0x43c1ffff | aes\_ip@43c10000 |
| desip\_0 | 0x43c20000 | 0x43c20000-0x43c2ffff | des\_ip@43c20000 |
| gcdip\_0 | 0x43c30000 | 0x43c30000-0x43c3ffff | gcd\_ip@43c30000 |

7.3 device tree compiler

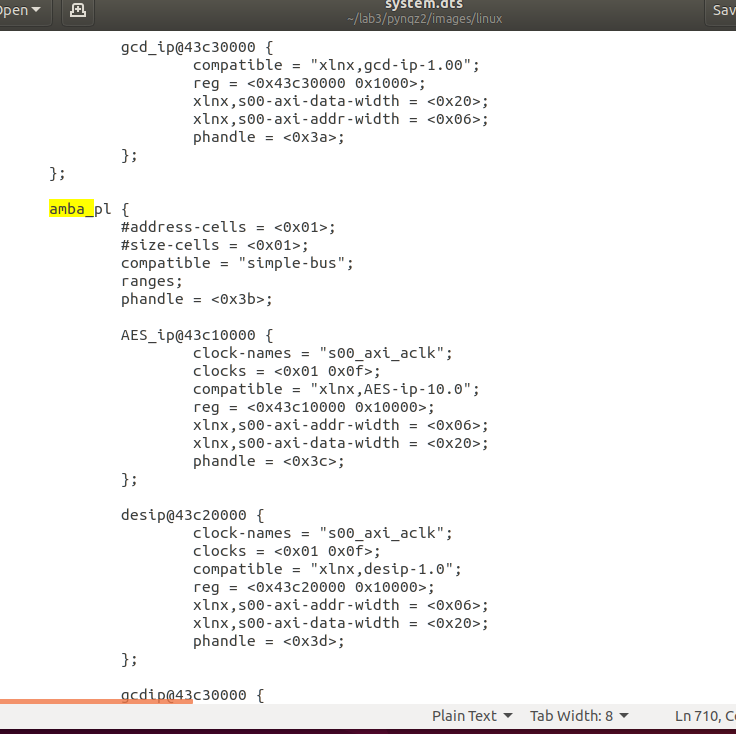


圖7.3.1 system.dts檢查模組匯入

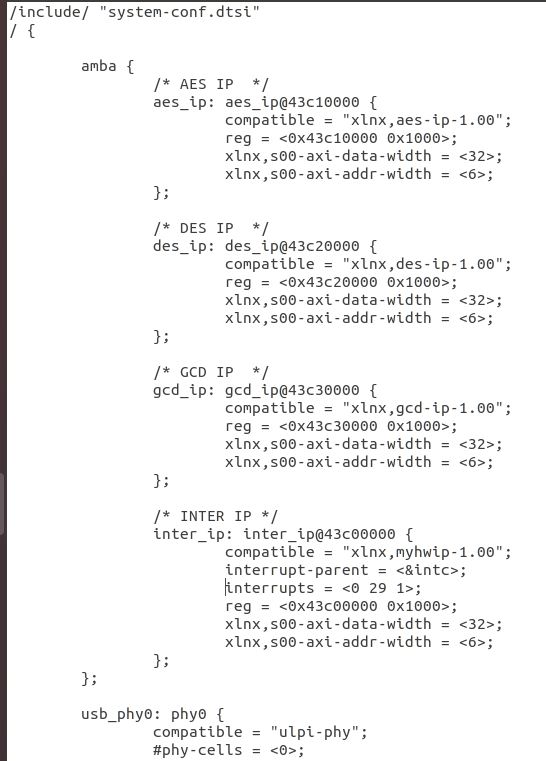


圖7.3.2 system-conf.dtsi編譯前修改

7.4 komod test result

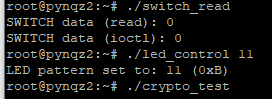
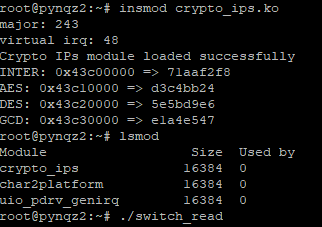


圖7.4.1 模組載入資訊、switch按鈕值、寫入led燈值

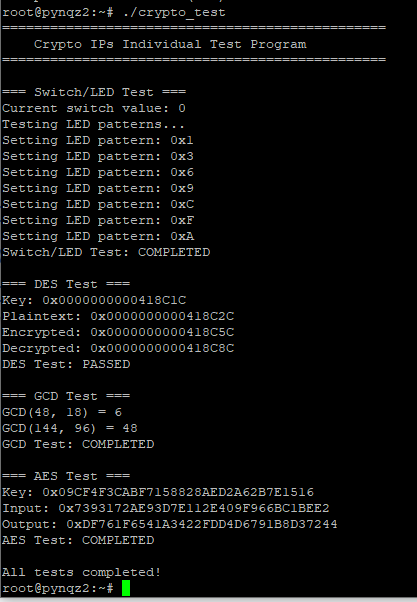


圖7.4.2 ./crypto\_test腳位以及功能測試

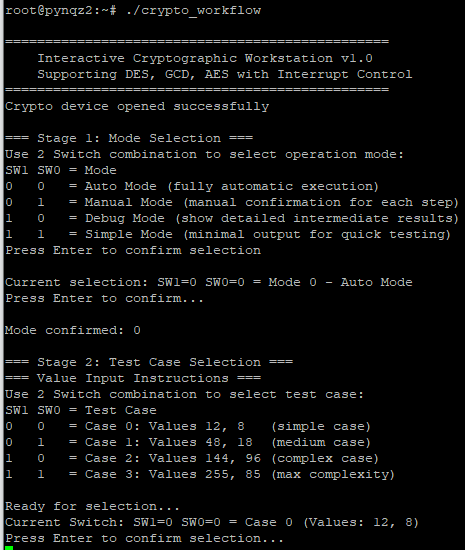


圖7.4.3 ./crypto\_workflow(Enter下一個stage)

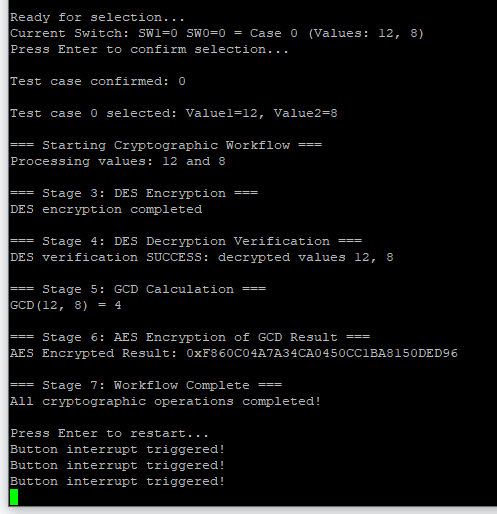


圖7.4.4 ./crypto\_workflow(完成以及最後push button中斷)

## **8.Interrupt**

* 流程簡介：用戶選擇模式（Switch Button） → 選擇測試資料（Push Button） → DES 加密 → DES 解密 → GCD 計算 → AES 加密
* 用戶模式選擇：

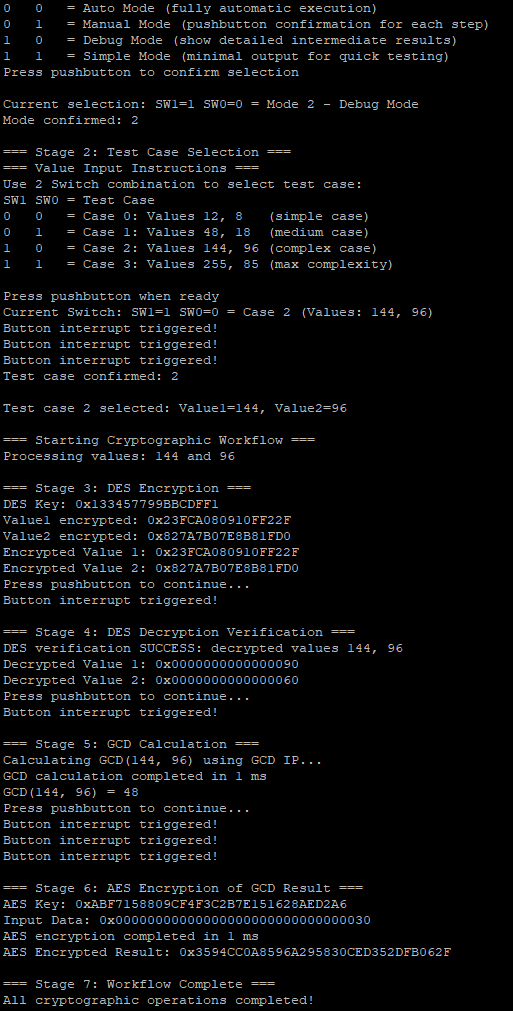
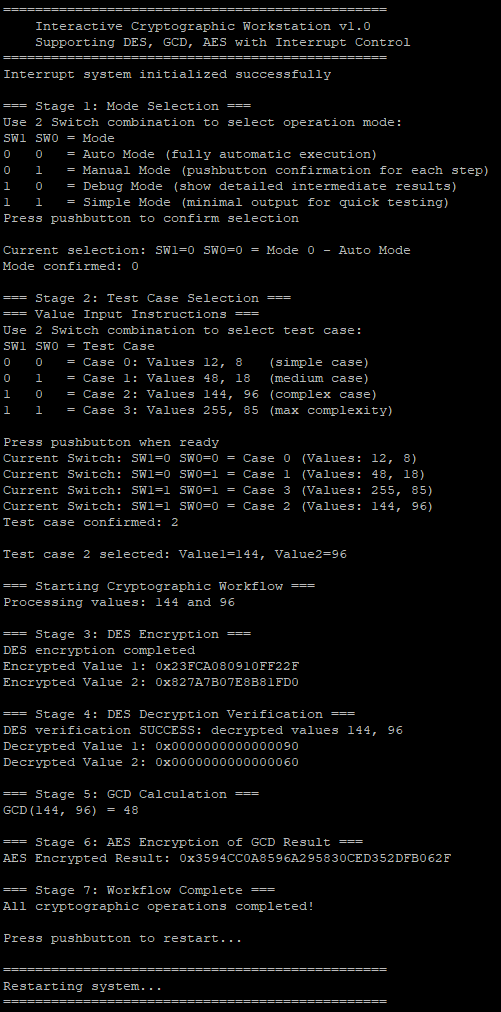
| 值（Switch Button) | 模式 | 說明 |
| --- | --- | --- |
| 00 | Auto | 每步驟自動進行 |
| 01 | Manual | 需按按鈕才能進入下一步 |
| 10 | Debug | 輸出完整細節資料 |
| 11 | Simple | 僅顯示主要結果 |

* 選擇測試資料：

|  |  |  |
| --- | --- | --- |
| 值（Push Button) | 測使案例 | 說明 |
| 00 | case 0 | 12,8 |
| 01 | case 1 | 48,18 |
| 10 | case 2 | 144,96 |
| 11 | case 3 | 255,85 |

* LED燈號顯示：

|  |  |
| --- | --- |
| 數值 | 狀態 |
| 0001 | Idle |
| 0011 | Input |
| 0110 | DES 工作中 |
| 1001 | GCD 計算中 |
| 1100 | AES 加密中 |
| 1111 | 工作完成 |



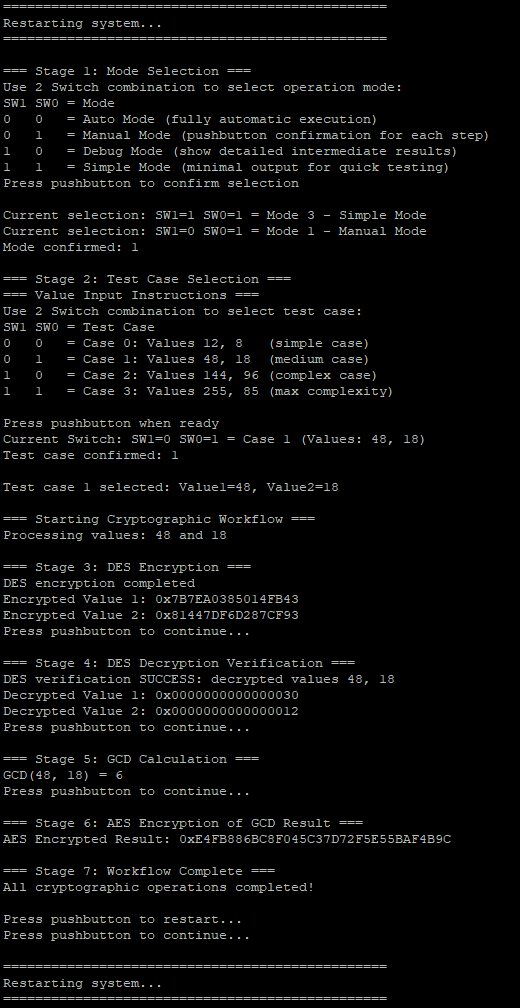
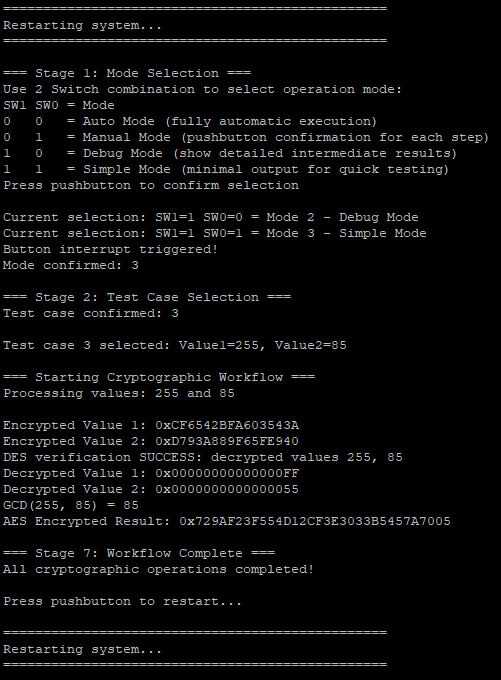


圖8.1.1 Interrupt輸出結果

# 分工表

|  |  |
| --- | --- |
| IP | |
| AES ip | 林庭毅 |
| GCD ip | 陳冠維 |
| DES ip | 陳冠維、羅豐祥 |

|  |  |
| --- | --- |
| IP 結合功能 | |
| 電路整合 | 林庭毅 |
| Standalone | 陳冠維、陳祥鈞 |
| FreeRTOS | 陳冠維、陳祥鈞 |
| Linux\_Driver | 林庭毅 |
| Interrupt | 林庭毅 |

|  |  |
| --- | --- |
| 文書處理 | |
| 排版 | 陳祥鈞 |
| 簡報 | 羅豐祥 |