# 中原大學電子工程學系

# 專題實作進度報告

題目：記憶體修復資料的無失真壓縮

Lossless Data Compress for Memory Hard Repair(iSTART-TEK INC.)

專題生：10926125 簡紹軒

10926113 吳東龍

10926230 蔣亦晴

指導教授：黃世旭 教授

指導學長姊：游雅程學姊、林永婕學姊

中華民國 112 年 9 月 11 日

# 專題公開展示會申請表

## 112 學年度電子系專題公開展示會申請表

| | | |
|---|---|---|
| 組別 | 晶片設計第1組 （半導體第1組、晶片設計第1組、通訊系統第1組……） | |
| 題目 | 記憶體修復資料的無失真壓縮<br>Lossless Data Compress for Memory Hard Repair (iSTART-TEK INC.) | |
| 摘要 | 硬修復是在執行記憶體測試後將關於錯誤位元的資訊及重新分配後的資訊儲存於 efuse 或是 OTP 中。由於其非揮發性，因此可減少系統啟動時間。然而因在 IC 中記憶體數量成長，如何降低 efuse/OTP 面積成本且保證記憶體正確性成了難題。若能將需存儲的資料進無失真資料壓縮就可達到降低 efuse 面積的要求。 | |
| 組長 | 姓名 | 簡紹軒 |
| | 班級及學號 | 電子四甲 109261125 |
| | 手機及E-mail | 0919577781 a0919577781@gmail.com |
| | 負責內容 | ~~a0919577781@gmail.com~~ 找資料、譯篇壓縮耗式提供想法 |
| 組員 | 姓名 | 蕭永晴 |
| | 班級及學號 | 電子四乙 109261230 |
| | 手機及E-mail | 0908159342 lz198935@gmail.com |
| | 負責內容 | 找資料、主要寫解壓縮耗式、提供想法 |
| 組員 | 姓名 | 吳秉龍 |
| | 班級及學號 | 電子四甲 109261137 |
| | 手機及E-mail | 0966482437 poinytrewq020p@gmail.com |
| | 負責內容 | 找資料、補耗式不足之處 |
| 組員 | 姓名 | |
| | 班級及學號 | |
| | 手機及E-mail | |
| | 負責內容 | |
| 指導老師簽名 | 黃 | |

*本表為專題報告指定上傳檔案，請依規定放入報告，方完成報告繳交。

## 摘要

隨著積體電路(Integrated Circuit)的蓬勃發展，嵌入式記憶體無論是在數量或是密度上都以驚人的速度在成長，而記憶體的半導體製程中有可能因外在因素導致記憶體發生缺陷(Defect)，針對此種情況通常會在記憶體電路中加入記憶體內建式自我測試電路(Memory Built-In Self Test, MBIST)達成記憶體自我測試目的。

MBIST 電路能藉由有效的演算法檢測記憶體內部是否有存在任何形式的故障，在設計中加入記憶體的內建自我修復機制(Built-In Self Repair, BISR)則可避免由記憶體故障導致的資料遺失，該方法被設計來分配用於修復錯誤位元的備用記憶體，在記憶體修復機制中，又分為軟修復(Soft Repair)與硬修復(Hard Repair)。

其中硬修復則是在執行記憶體測試後將關於錯誤位元的資訊以及重新分配後的資訊儲存於 eFuse 或是 OTP(One Time Programmable)中。由於其非揮發性，因此可以減少系統啟動的時間。然而因在 IC 中記憶體的數量成長，如何在降低 eFuse/OTP 面積成本且同時保證記憶體的正確性就成為了一個難題。若是能先將需存儲的資料進行無失真的資料壓縮(Lossless Data Compression)，就可達到降低 eFuse 面積的要求。



**關鍵字：記憶體內建式自我測試電路、無失真資料壓縮壓縮**

# Abstract

With the booming of integrated circuits, embedded memory is growing at an alarming rate in terms of quantity and density. Memory Built-In Self Test (MBIST) is usually added to the memory circuit for the purpose of memory self test.

The MBIST circuit detects the presence of any form of internal memory failure by means of an efficient algorithm. The design incorporates built-in self-repair (BISR) of the memory to prevent data loss due to memory failure, which is designed to allocate spare memory for the repair of faulty bits. Among the memory repair mechanisms, there are soft repair and hard repair.

The hard repair is to store the information about the erroneous bits and the reallocated information in eFuse or OTP (One Time Programmable) after the memory test. Because of its non-volatile nature, it can reduce the system startup time. However, as the amount of memory in ICs grows, it is a challenge to reduce the cost of eFuse/OTP area and to ensure the correctness of memory at the same time. If the data to be stored can be compressed without distortion (Lossless Data Compression) first, the requirement of eFuse area can be reduced.



**Keywords : Memory Built-In Self Test, lossless data compression**

# 目錄

# 第一章 緒論

## 1-1. 研究背景

隨著積體電路的迅猛發展，嵌入式記憶體在數量和密度上持續增長。然而，在半導體製程中，外部因素可能導致記憶體出現缺陷。為了應對這種情況，通常在記憶體電路中加入記憶體內建式自我測試電路（MBIST），以實現記憶體的自我測試。MBIST 的設計除了要能夠檢測各種記憶體故障之外，還需要在節省電路面積、降低功率消耗和減少測試時間方面有所考慮。

同時，內建自我修復（BISR）機制的加入可以避免由記憶體故障導致的資料丟失。BISR 機制通過使用備用記憶體和內建冗餘分析（BIRA）來實現修復功能，即分配備用記憶體來修復錯誤位元。

在記憶體修復機制中，又分為軟修復(Soft Repair)與硬修復(Hard Repair)，其中硬修復則是在執行記憶體測試後將關於錯誤位元的資訊以及重新分配後的資訊儲存於 eFuse 或是 OTP(One Time Programmable)中。由於其非揮發性，因此可以減少系統啟動的時間。然而非揮發性記憶體所帶來的面積劣勢以及額外的硬體成本花費，都是需要被審慎考量的問題。

## 1-2. 研究動機

eFuse 與 OTP 型態的記憶體由於其非揮發性的特性，被用來儲存記憶體錯誤的位址以及資料。然而由於在 IC 中記憶體的數量成長，因此如何在降低 eFuse/OTP 面積成本，且同時保證記憶體的正確性就成為了一個難題。若是能夠先將需要存儲的資料進行無失真的資料壓縮(Lossless Data Compression)，就可以達到降低 eFuse 面積的要求。

# 第二章 演算法規劃

## 2-1. 演算法摘要

這是一種游長縮減的二元序列壓縮編碼方法，0、1 游長意思為連續有多少個 0 或 1 的數值，首先將待壓縮之 0、1 二元字串序列轉換為游長序列，大於 1 之游長以持續除 2 來壓縮每段 0、1 游長至游長數為 1 為止，則會產生 01 相間的新游長序列，在壓縮過程中的餘數則被儲存至額外的檔案方便解壓縮時使用，而 01 相間之新游長則會計算重複次數進行壓縮，最後生成一串關係碼來辨別為 0 或 1 開頭及新游長壓縮前有無多一位和 01 重複次數。

該方法能避免對較短游長重新編碼而導致編碼相較未壓縮之編碼位元相差不大甚至位元更多，能較大幅度的壓縮及讀檔。

## 2-2. 演算法步驟

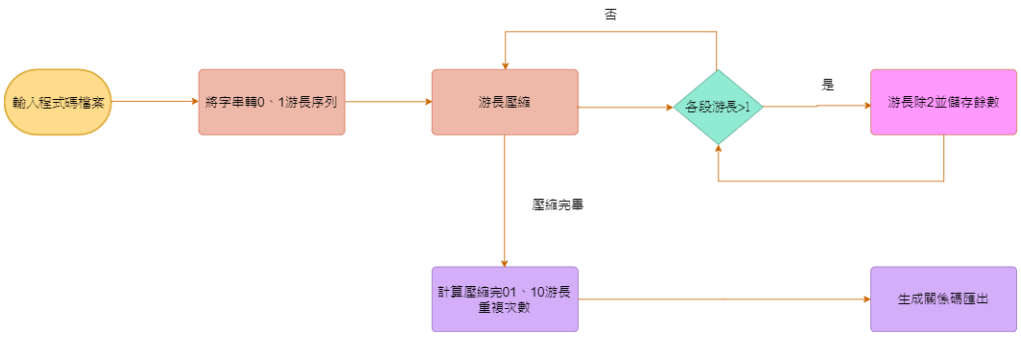
將題目給予的 01 字串轉化成游長序列，將每段大於 1 的 0、1 游長持續除 2 的餘數 0 和 1 來生成各段游長的關係碼，若關係碼為 01 則代表此段游長為 6，為 00 則為 4，命名方式為第幾段的游長，最後剩餘的 01 相間的新游長則去判斷首位為 0 或 1，並且在計算重複次數後儲存是否多出 1 位，最後產出之關係碼首位則代表 01

或 10 開頭，二位為 0 代表無多出 1 位而反之則有，其餘數字為 01 或 10 之重複次數以二進位制儲存。

解壓縮則先判斷前兩位數字，再將其餘數字轉成十進位制擴充 01 或 10 游長，再配合各段餘數之關係碼還原，若無餘數關係碼則不做還原動作即可。

# 第三章 演算法成果

## 3-1. 演算法運行




上圖為壓縮流程圖

下為壓縮程式虛擬碼

1. 先計算原二進制資料的各 0 和 1 游長長度並記錄在 vec

宣告記錄 0 和 1 游長的 vector: vec;

void computeLength(string str: 要被壓縮的二進制資料){

int zero = 0;

int one = 0;

while(str 尚未被遍歷完) do

while(str 當前位置為 0) do

zero++;

if(str 當前位置不為 0) then

將 zero 記錄在 vec;

```
        while(str 當前位置為 1) do
            one++;
        if(str 當前位置不為 1) then
            將 one 記錄在 vec;
}
```

2. 開始壓縮，將各 0 和 1 游長長度壓縮到 1 為止，並記錄餘數(關係碼)

```
for(i=1 to vec.size) do {
    if(vec[i] == 1)  跳過不處理;
    while(vec[i] > 1) {
        vec[i] = vec[i] / 2;
        記錄除以 2 之後的餘數;
    }
}
```

3. 壓縮結束，因為把游長長度壓到 1，資料成 10 或 01 交錯，所以計算有幾組交錯，再轉成二進制記錄下來

bin: 第一個 bit 代表開頭為 0 或 1，第二個 bit 代表是否多出一個 bit，後面的 bits 代表有幾組 10 或 01 交錯

string computeGroupAnd2Binary(string compressed: 壓縮後字串){

宣告字串代表二進制字串: string bin;

if(compressed 開頭為 1) then

bin.append("1");

if(compressed 結尾為 0，剛好 10 配對完) then

bin.append("0");

記錄總共有 compressed.size()/2 組 10 交錯

else(compressed 結尾為 1)

bin.append("1");

記錄總共有 compressed.size()/2 組且最後多出一個 bit

else(compressed 開頭為 0)

if(compressed 結尾為 1，剛好 01 配對完) then

bin.append("0");

記錄總共有 compressed.size()/2 組 01 交錯

else(compressed 結尾為 0)

bin.append("1");

記錄總共有 compressed.size()/2 組 01 交錯且多出一個 bit

string binary = 10 或 01 交錯組數之二進制表示;

bin.append(binary);

return bin;

}

4. 將關係碼和壓縮後字串輸出成二進制檔

void writeBinaryFile(string input: 要寫入的資料

string filePath: 寫入檔案的路徑

int i: -1 代表寫入的是關係碼，其他的是二進制字串)

{

宣告寫入檔案的 stream: ofstream outfile;

outfile.open(指定路徑 filePath);

if(outfile 開啟成功) then

if(i == -1) then

寫入二進制字串;

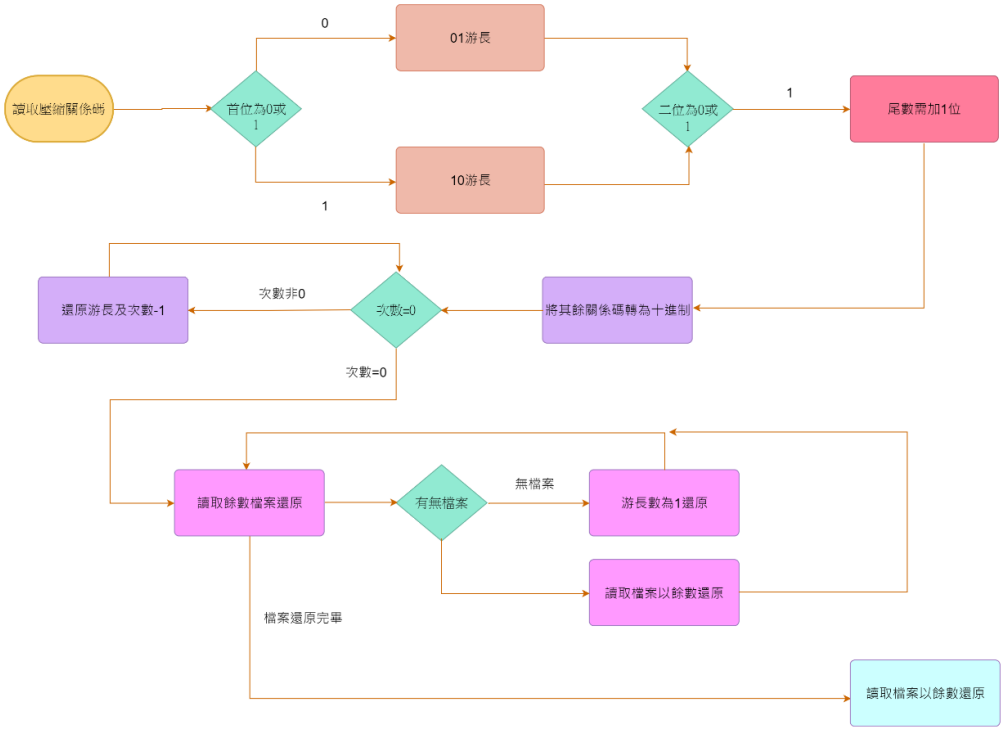
else 寫入關係碼;

else 開啟失敗

exit(1);

outfile.close();

}




上圖為解壓縮流程圖

下為解壓縮程式虛擬碼

1.解析當前 cpp 檔的位置，以便讀取壓縮資料夾中的關係碼檔和二進制檔來解壓縮

```
string currentPath = exec("pwd"); //執行 pwd command
string rootPath = currentPath.parent_path();
string filePath = rootPath +  解壓縮資料夾名稱  +  解壓縮資料檔名;
```

其中 exec 函式如下

```
string exec(const char* cmd:  要執行的 command){
    宣告 buffer 用來存執行 cmd 的輸出;
    FILE* pipe = popen(cmd, "r");
    if(!pipe) then  丟出錯誤;
    try{
        while(當 buffer 尚未讀取結束) do
            result += buffer;
    }
    catch(){
        pclose(pipe);
```

```
            丟出錯誤;
        }
        pclose(pipe);
        return result;
}
```

2.解析二進制檔的內容，將解析結果存入 base 變數

根據 filePath 找到二進制檔後，將資料存入 str 變數;

```
if(str 的第一個 bit 為 1) then
        check = "10"; //代表資料為 10 交錯;
else
        check = "01"; //代表資料為 01 交錯;

if(str 的第二個 bit 為 1) then
        last = 1; //代表資料配對到最後多出一個 bit
else(str 的第二個 bit 為 0)
        lase = 0; //代表資料剛好配對完
```

將 str 其他後面的 bits 存入 compressed 變數;

int decimal = compressed 轉十進位; //代表有幾組 10 或 01 交錯

for(int i=0 to decimal) do

    base += check;

if(last == 1，最後多出一個 bit) then

    if(base 為 10 交錯) then

        base += '1';

    else(base 為 01 交錯) then

        base += '0';

3.利用 base 變數和壓縮生成的關係碼檔來解壓縮，並將解壓縮結果存入 bin 變數

for(int i=0 to base.length()) do

    讀取關係碼檔內容，並存入 r 變數;

    if(此段游長有關係碼) then

        for(int i=r 的最後一個 bit to 0) do

            將游長值乘以 2 再加上 r[i];

4.將解壓縮結果以 byte 為單位存起來

string binaryData;

for(int i=0 to bin.length(); i+=8) do

    string byteData =  擷取 bin 中第 i 到 i+8 位置的子字串;

    將 byteData 轉成 char，存入 byte 變數;

    binaryData.push_back(byte);

5.將處理完的資料輸出成二進制檔

ofstream outfile(輸出檔名, 以 binary 方式輸出);

if(outfile 開啟成功) then

    outfile.write(binaryData.c_str(), binaryData.size());

    outfile.close();

else(outfile 開啟失敗) then

    cerr << "Error opening the file" << endl;

# 3-2. 運行程式成果

測資一

```
Compress ProblemE_Test_Case_1_20230703...

Compressed ProblemE_Test_Case_1_20230703 result:
Relation code size: : 311036
Non-relation code size: 11111100000011110000

Execution time: 12.274 s
Original size: 983040
Compressed Data size(compressed string + relation code): 18 + 311036 = 311054
```

```
stevenchien@ubuntu:~/Desktop/test/Decompress$ ./cade0042.exe ProblemE_Test_Case_1_20230703.bin
Decompressing ProblemE_Test_Case_1_20230703.bin...
Binary data has been written


Decompression succeeded.

Execution time: 17.1947 s
```

測資二

```
Compress ProblemE_Test_Case_2_20230703...

Compressed ProblemE_Test_Case_2_20230703 result:
Relation code size: : 269478
Non-relation code size: 10110100000001111101

Execution time: 4.33382 s
Original size: 851968
Compressed Data size(compressed string + relation code): 18 + 269478 = 269496
Compressed ratio(oldSize/newSize): 3.16134
```

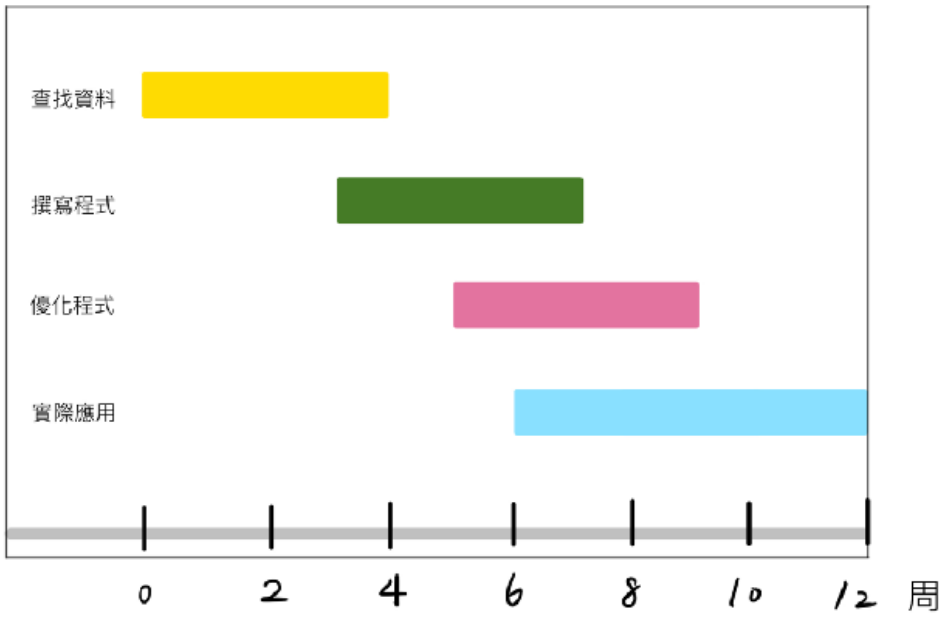
```
stevenchien@ubuntu:~/Desktop/test/Decompress$ ./cade0042.exe ProblemE_Test_Case_2_20230703.bin
Decompressing ProblemE_Test_Case_2_20230703.bin...
Binary data has been written


Decompression succeeded.

Execution time: 14.2821 s
```

測資三

```
Compress ProblemE_Test_Case_3_20230703...

Compressed ProblemE_Test_Case_3_20230703 result:
Relation code size: : 165764
Non-relation code size: 0011111111110110111

Execution time: 6.25381 s
Original size: 524288
Compressed Data size(compressed string + relation code): 17 + 165764 = 165781
Compressed ratio(oldSize/newSize): 3.16253
```

```
stevenchien@ubuntu:~/Desktop/test/Decompress$ ./cade0042.exe ProblemE_Test_Case_3_20230703.bin
Decompressing ProblemE_Test_Case_3_20230703.bin...
Binary data has been written


Decompression succeeded.

Execution time: 8.81643 s
```

# 第四章 結論與展望




12 周進度規畫表之甘特圖

由於 IC 中記憶體數量成長，若先將需存儲的資料進行無失真的資料壓縮，可達到降低 eFuse 面積的要求，進而降低 eFuse 成本並保持資料完整性，由於 0、1 數據的變化性可大可小，有可能在壓縮時讓位元變多或可能壓縮時間過長，種種限制也讓這份挑戰增添了許多可能性。

因此我們在程式中加入了判斷的部分，令壓縮檔不大於原檔，將每段游長之關係碼額外儲存方便區分檔案，並利用前兩位判別前綴碼與壓縮至最底之游長重複次數之二進位數值產生一串新關係碼，讓檔案不僅有效壓縮且時間快速，也方便了解關係碼之意義。

# 參考資料

[1] 游长缩减的二元序列压缩编码方法

https://patents.google.com/patent/CN102651795B/zh

[2] Huffman Coding 霍夫曼編碼. Huffman Coding 編碼、解碼，以及 Huffman Tree 的建立。 | by bhch3n | Medium

https://medium.com/@bhch3n/huffman-coding-%E9%9C%8D%E5%A4%AB%E6%9B%BC%E7%B7%A8%E7%A2%BC-3879df5ecddc

[3] process - How do I execute a command and get the output of the command within C++ using POSIX? - Stack Overflow

https://stackoverflow.com/questions/478898/how-do-i-execute-a-command-and-get-the-output-of-the-command-within-c-using-po

[4] Linux 文字編譯器 nano 快捷鍵小抄-黑暗執行緒 (darkthread.net)

https://blog.darkthread.net/blog/nano-shortcut/

[5] In C++, what's the fastest way to tell whether two string or binary files are different? - Stack Overflow

https://stackoverflow.com/questions/15118661/in-c-whats-the-fastest-way-to-tell-whether-two-string-or-binary-files-are-di

[6] README 寫法 | QWERTY (gitqwerty777.github.io)

http://gitqwerty777.github.io/art-of-readme/

# 專題生簡介


姓名：吳東龍

籍貫：中華民國

生日：90/10/09

學歷：台南市

班級：電子四甲

學號：10926113

中原大學電子工程學系

姓名：簡紹軒

籍貫：中華民國

生日：91/06/18

學歷：新北市

班級：電子四甲

學號：10926125

中原大學電子工程學系

姓名：蔣亦晴

籍貫：中華民國

生日：90/11/22

學歷：桃園市

班級：電子四乙

學號：10926230

中原大學電子工程學系

# Turnitin 專題報告比對結果確認書

Turnitin 專題報告比對結果確認書

專題組別：晶片設計第1組（半導體第1組、晶片設計第1組、通訊系統第1組……）

專題生姓名1：簡紹軒　（學號：109261275）

專題生姓名2：蔣亦晴　（學號：109261230）

專題生姓名3：吳東龍　（學號：109261117）

專題生姓名4：　（學號：　）

訂於 112-1 學期繳交專題報告，

中文報告名稱：「記憶體修復資料的無失真壓縮」

英文報告名稱：「Lossless Data Compress for Memory Hard Repair」
(iSTART-TEK INC)

目前已完成專題報告初稿比對，列印結果報告送交指導老師並確認無抄襲情事，此致

中原大學電子系。

專題生蔣亦晴 吳東龍 簡紹軒（簽名）

※組內每位專題生皆需親自簽名，不可代表※

指導老師 黃[illegible]（簽名）

*本表為專題報告指定指定上傳檔案，請依規定放入報告中，方完成報告繳交。

# 電子系專題進度報告繳交確認表

## 電子系專題報告繳交確認表(112-1 學期)

| 勾選 | 格式確認項目 | 備註說明 | 應繳日期 |
|---|---|---|---|
| ☑ | 字體：一般內文請採用 14-point 標楷體（中文）及 Times New Roman （英文、阿拉伯數字及符號等）；各層級的標題採用 16 或 18-point 的粗體字（boldface typed）。 | 電子檔格式及內容確認 | |
| ☑ | 電子檔第 1 頁<br>-專題報告封面 | 請依頁數順序排列 | |
| ☑ | 電子檔第 2 頁<br>-專題公開展示會申請表 | 需給**指導老師簽名或簽章**後放入電子檔 | |
| ☑ | 電子檔第 3 頁<br>-中文摘要 | | |
| ☑ | 電子檔第 4 頁<br>-英文摘要 | | |
| ☑ | 電子檔第 5 頁<br>-目錄 | | |
| ☑ | 電子檔最後章節<br>-結論與展望 | | |
| ☑ | 電子檔最後附錄<br>-參考資料及專題生簡介 | | |
| ☑ | 電子檔倒數第 2 頁<br>專題報告完稿進行 Turnitin 專題報告相似度比對<br>-比對結果報告，請交給指導老師確認。確認後，請老師於「Turnitin 專題報告比對結果確認書」簽名或簽章。 | Turnitin 專題報告比對結果確認書：<br>**指導老師簽名或簽章**後放入電子檔 | |
| ☑ | 電子檔最末頁<br>-專題報告繳交確認表 | 請務必逐項勾選確認表單後放入電子檔 | |
| ☑ | **上述白底項目勾選確認後，**請確認下列說明之電子檔格式及檔名，將電子檔上傳至 i-learning。<br>1.上傳電子檔，請以 **PDF 檔上傳**<br>2.上傳電子檔，檔名：<br>**半導體/晶片設計/通訊系統第 1 組_10426101.10426202**<br>(按甲.乙.丙班及學號由小至大排序) | 電子檔上傳確認 | **9 月 12 日(二)**<br>**上午 11 時前繳交**<br>**不得遲交!!** |
| ☑ | **僅保留最新的專題報告電子檔**<br>(若重新上傳電子檔，請將舊的電子檔刪除。) | | |

※全部組員(含組長)學號：10926113，10926134，10926230

※以上確認項目由各組專題組長負責勾選確認，全部勾選後請組長簽名：簡紹軒

※專題組別：晶片設計第 1 組 (半導體第 1 組、晶片設計第 1 組、通訊系統第 1 組……)

**※報告不接受補交，請同學務必準時完成線上繳交專題報告，謝謝！**