

# 2018 IC Design Contest Preliminary

## 研究所組標準元件數位電路設計

### Huffman Coding

#### 1. 問題描述

赫夫曼編碼(Huffman Coding)是一種用於無損壓縮的可變長度編碼演算法，由 David Albert Huffman 於 1952 年發明(參考圖 1)。本題要請各位參賽者完成 Huffman Coding 的編碼產生器(huffman 電路)，輸入為原始資料，參賽者應該就輸入資料進行統計，並依統計結果產生對應的 Huffman Code 作為輸出，細節描述於後 2.3 系統功能描述一節。

本電路各輸入輸出信號的功能說明，請參考表 1。本電路系統方塊圖請參考圖 2。每個參賽隊伍必須根據下一節所給的設計規格及附錄 E 中的測試樣本完成設計驗證。

本次 IC 設計競賽比賽時間為上午 08:30 到下午 08:30。當 IC 設計競賽結束後，CIC 會根據第 3 節中的評分標準進行評分。為了評分作業的方便，各參賽隊伍應參考附錄 C 中所列的要求，附上評分所需要的檔案。軟體環境及設計資料庫說明請參考附錄 A 與附錄 B。

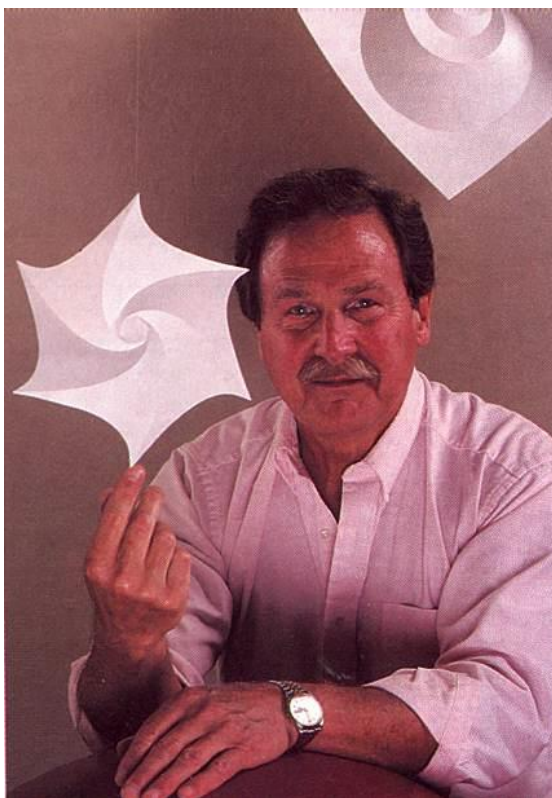


圖 1. David Albert Huffman，Huffman coding 的發明者  
(source: <https://www.huffmancoding.com/my-uncle/scientific-american>)

## 2. 設計規格

請注意：

1. 此次top module名稱及檔案名稱、大小寫須完全符合附錄B規範，若有引入其它模組或檔案請自行寫在設計檔內，測試檔不允許任何修改否則不予計分。
2. 最後評分方式為使用最後上傳檔案版本評分，並以最後上傳檔案版本時間為依據，請參考3.評分標準。

### 2.1 系統方塊圖

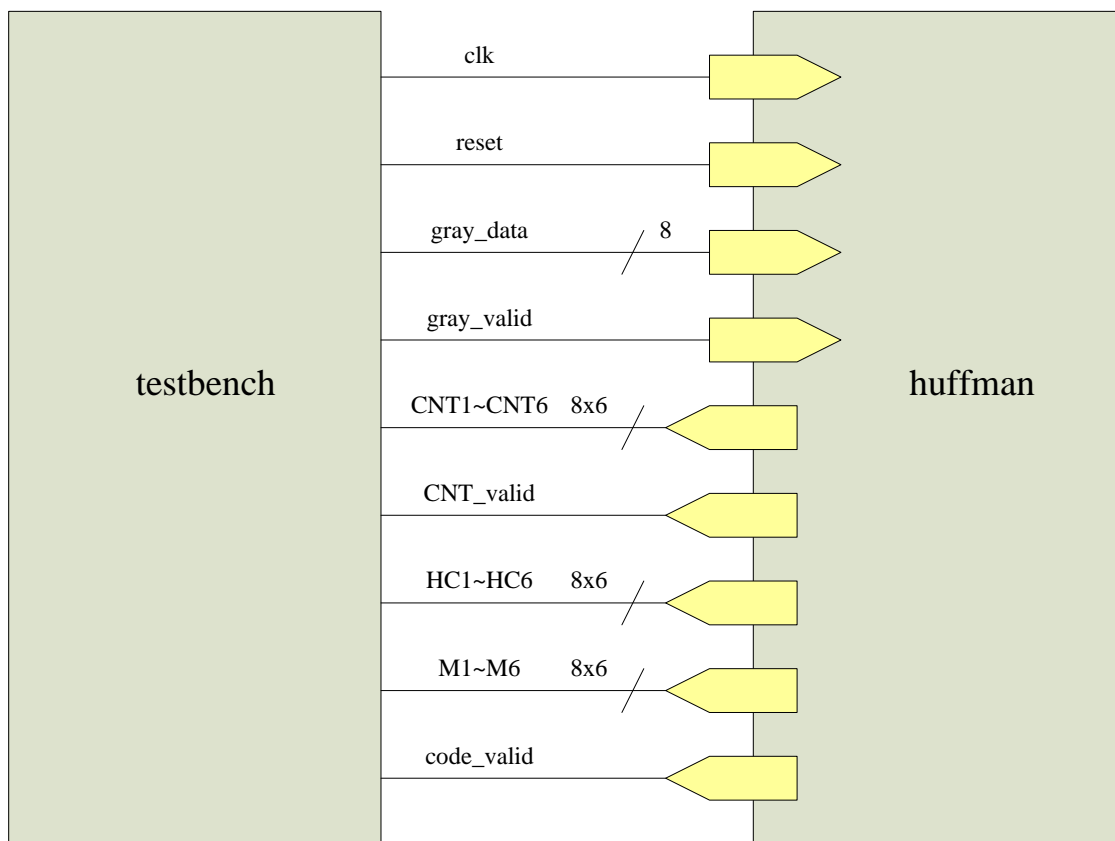


圖 2. 系統方塊圖

## 2.2 輸出入訊號描述

表一、輸入/輸出信號

| Signal Name | I/O | Width | Simple Description  |
|-------------|-----|-------|---|
| clk         | I   | 1     | 本系統為同步於時脈正緣之同步設計。   |
| reset       | I   | 1     | 高位準”非”同步(active high asynchronous)之系統重置信號。  |
| gray_data   | I   | 8     | 灰階圖像資料匯流排。Testbench 會在 gray_valid 有效期間每 cycle 送出一筆灰階圖像資料。   |
| gray_valid  | I   | 1     | 灰階圖像資料指示信號。當為 High 時，表示 gray_data 上的資料是有效的。   |
| CNT1~CNT6   | O   | 8x6   | Huffman 電路對輸入階段的灰階圖像資料進行統計的統計結果。其中 CNT1 代表 A1 的數目，CNT2 代表 A2 的數目，其餘類推。CNT1~CNT6 各用 8-bit 資料來紀錄資料，這些資料在 CNT_valid 訊號為 High 時，必須正確顯示各 symbol 的統計數目。                               |
| CNT_valid   | O   | 1     | 各 symbol 統計數目 CNT1~CNT6 輸出有效通知信號。當輸入資料完成統計後，此訊號拉為 High 並維持一 cycle，該 cycle 內的 CNT1~CNT6 分別代表各 symbol 的數目。  |
| HC1~HC6     | O   | 8x6   | 經 huffman 電路所決定出各 symbol 的 huffman 編碼內容。其中 HC1 到 HC6 均為 8-bit 資料寬度，分別代表本題目中的六個 symbol A1~A6 的 Huffman code。完整的 Huffman code 須由每個 symbol 的編碼內容及對應的遮罩來表示(參考 2.3.1 節)。             |
| M1~M6       | O   | 8x6   | 經 huffman 電路所決定出各 symbol 的 huffman 編碼遮罩(mask)。每個遮罩均為 8-bit 寬度，其中 M1 代表 A1 的遮罩，M2 代表 A2 的遮罩，其餘類推。完整的 Huffman code 須由每個 symbol 的編碼內容及對應的遮罩來表示(參考 2.3.1 節)。                        |
| code_valid  | O   | 1     | huffman 電路產生出輸入灰階圖像資料對應的 huffman code 後的 HC1~HC6 及 M1~M6 輸出有效通知信號。當 huffman 電路完成產生 huffman code 後，此訊號拉為 High 並維持一 cycle，該 cycle 內的 HC1~HC6 及 M1~M6 分別代表各 symbol 的 Huffman code。 |

## 2.3 系統功能描述

考慮一張只有 100 pixel 的灰階圖片，每個 pixel 以 8 bit 儲存其灰階資料。此處為簡便起見只考慮六種灰階 A1、A2、A3、A4、A5、A6，其對應的數值(16 進位)如下表所示。在此定義 A1、A2、A3、A4、A5、A6 等 symbol 的索引值(index)分別為 1、2、3、4、5、6。

| Symbol | A1   | A2   | A3   | A4   | A5   | A6   |
|--------|------|------|------|------|------|------|
| Value  | 0x01 | 0x02 | 0x03 | 0x04 | 0x05 | 0x06 |

本電路功能當 *reset* 結束後，Host 端會將 *gray\_valid* 訊號拉為 High 表示此時每一個 cycle 的 *gray\_data* 均代表一個 pixel 的輸入，總共有 100 個 pixel。這 100 筆資料輸入後，*gray\_valid* 訊號拉為 Low。此為第一階段，系統電路應該就這些輸入對本次要進行處理的圖形資料進行各 symbol 發生次數的統計，並加以排序以得到此圖中各 symbol 的發生機率大小以產生 Huffman code。


| Symbol | P(Ai) |
|--------|-------|
| A1     | 0.1   |
| A2     | 0.4   |
| A3     | 0.06  |
| A4     | 0.1   |
| A5     | 0.04  |
| A6     | 0.3   |

圖 3. 各 symbol 發生機率，P(Ai)代表 Ai symbol 的發生機率

要解釋 Huffman coding 的最好方式莫過於實際舉例說明。故今假設統計出來 A1 的發生機率為 0.1(即 10%)，A2 發生機率為 0.4，A3 的機率為 0.06，A4 的機率為 0.1，A5 的機率為 0.04，A6 的機率為 0.3(如圖 3 所示)，並以此為例說明 Huffman coding 的產生方法，可分為 initialization、combination、split 三階段進行。

首先為 initialization 階段，將之各 symbol 發生機率依機率大小由大到小、從上而下排列下來；在 initialization 階段，若有機率相同者，則擁有較小索引值的 symbol 排上方(如 A1 跟 A4 的機率都是 0.1，A1 的索引值為 1 比較小，所以 A1 排在 A4 上方)，排成後如圖 4 所示

| Symbol | P(Ai) |
|--------|-------|
| A1     | 0.1   |
| A2     | 0.4   |
| A3     | 0.06  |
| A4     | 0.1   |
| A5     | 0.04  |
| A6     | 0.3   |



| P(Ai) | Symbol |
|-------|--------|
| 0.4   | A2     |
| 0.3   | A6     |
| 0.1   | A1     |
| 0.1   | A4     |
| 0.06  | A3     |
| 0.04  | A5     |

圖 4. 依 symbol 發生機率大小排列

接著會經過一連串 combination 階段，每一回合將前一回合中機率最低的兩組組合在一起，合併後的機率與其他組的機率再重新按機率大小排序(re-order)。所以第一次的 combination (C1 回合，如圖 5 所示)用的原始資料，目前 A3 的機率為 0.06，A5 的機率為 0.04，是現在機率最小的兩組，

故將兩組合併成一項{A3,A5}，合併後的項對應的發生機率為合併項個別機率的和，所以{A3,A5}的機率為  $0.06+0.04=0.1$ 。合併項的機率要再經過排序，因  $0.1$  比上面的 A4 發生機率( $0.1$ )是一樣的，此處規定在 combination 階段一樣機率者，統一規定合併項放在此相同機率的群組中的最後一位。右方為第一回合對應的 Huffman tree，Huffman tree 的畫法請參考附錄 F 中的介紹。

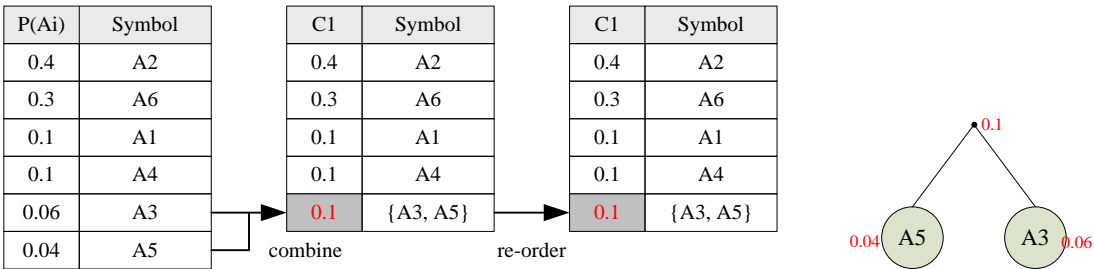


圖 5. 第一次組合 C1 回合

在 C1 回合後，進行 C2 回合(參考圖 6)，將 C1 回合中機率最低的兩組組合在一起，合併後的機率與其他組的機率再重新按機率大小排序(re-order)。所以 C2 回合會檢查 C1 回合結果中機率最小的兩組(即排在最下面的兩組)，分別是 A4(機率  $0.1$ )及{A3,A5}(機率  $0.1$ )，將兩組合併成一項{A4,A3,A5}，合併後的項對應的發生機率為合併項個別機率的和，所以{A4,A3,A5}的機率為  $0.1+0.1=0.2$ 。合併項的機率要再經過排序，此處合併項{A4,A3,A5}機率為  $0.2$ ，比 A1 的機率  $0.1$  大，但比 A6 的機率  $0.3$  小，故在 re-order 排序時，要將{A4,A3,A5}合併項位置移到 A1 及 A6 的中間，到此完成 C2 回合。

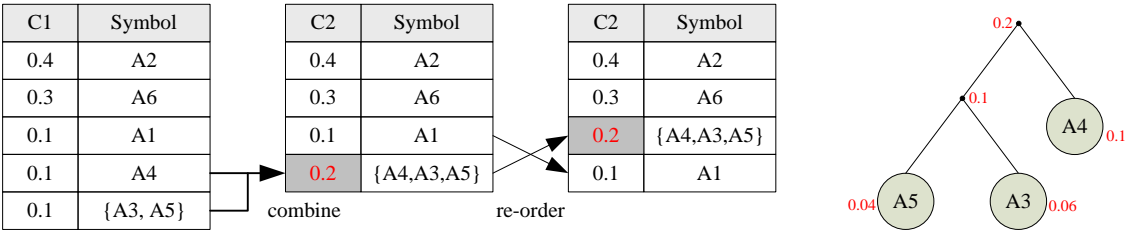


圖 6. 第二次組合 C2 回合

如上說明，combination 階段會一直進行合併到只剩兩項時停止，故還要經過兩次合併 C3、C4 回合(如圖 7、圖 8 所示)。

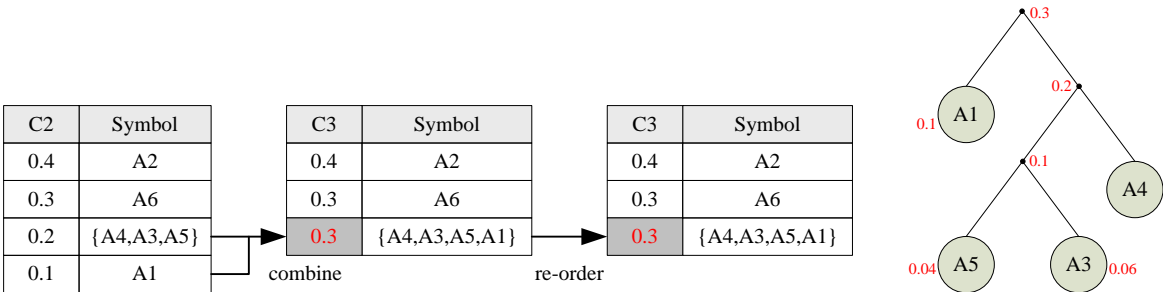


圖 7. 第三次組合 C3 回合

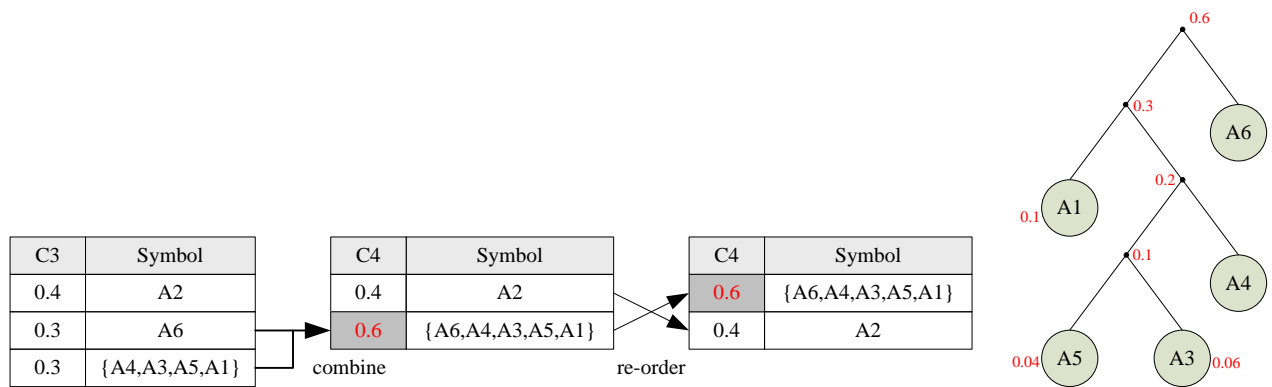


圖 8. 第四次組合 C4 回合

將所有的 combination 階段放在一起觀察，即圖 9 所示，有時也會將此資料製成如圖 10 的 Huffman table 來觀察(也有人是做成 Huffman Tree)。

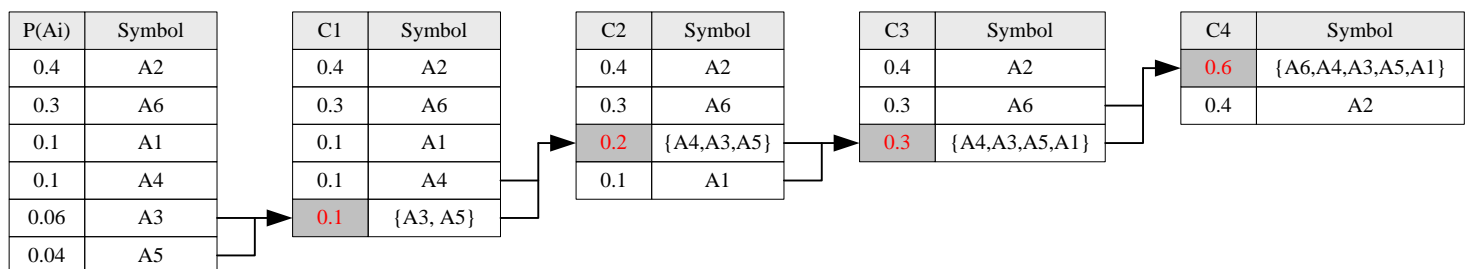


圖 9. 組合程序總覽

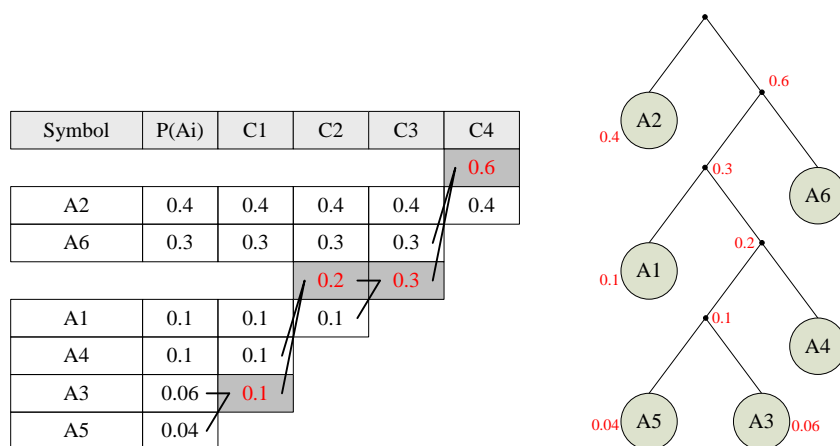


圖 10. 對應的 Huffman table

接著是一連串倒推回來的 split 階段，在此階段會逐步對各 symbol 賦予其 Huffman code。首先由 C4 的結果開始(參考圖 11)，C4 只有兩組 symbol，一組為排在上、擁有發生機率 0.6 的 {A6,A4,A3,A5,A1} 混合集團，一組是排在下、擁有發生機率為 0.4 的純血 A2，在本題目中規定上方的分配 code 0，下方的分配 code 1。所以 symbol A2 用 1 來表示其 Huffman code，同時 0 代表 {A6,A4,A3,A5,A1} 的混合集團。此即意味著當 decoder 讀到 1 即代表是一個 A2，但是讀到 0 時只知道是 A6、A4、A3、A5、A1 的其中一個，尚無法確認是那一個，所以對 {A6,A4,A3,A5,A1} 的編碼 0 需要追加額外的編碼，以便能確認是那一個 symbol。

| assign |                  |      |
|--------|------------------|------|
| C4     | Symbol           | code |
| 0.6    | {A6,A4,A3,A5,A1} | 0    |
| 0.4    | A2               | 1    |

圖 11. C4 回合的 Huffman code

接著由 C4 split 到 C3。每次 split 時均是 combination 階段每一次 combine 的逆推，並規定上方的分支追加編碼 0，下方的分支追加編碼 1(參考圖 12)。故 C4 回合中，混合集團{A6,A4,A3,A5,A1}會由原來的 combination 過程逆推 split 成 A6 及{A4,A3,A5,A1}兩項(如圖 13)，其中 A6 位於上方，故其編碼為 C4 中{A6,A4,A3,A5,A1}分配到的繼承編碼 0 及 split 時追加的編碼 0，故 A6 的 Huffman code 為 00。同理{A4,A3,A5,A1}的編碼為 01，但是因為{A4,A3,A5,A1}仍為混合集團，故須依照上面方式進一步的進行 split，直到每個單獨的 symbol 為止。完整的 split 過程請參考圖 14，也可參考利用 Huffman table 進行 split 的圖解(如圖 15)。

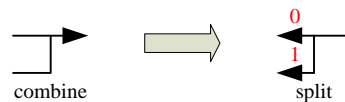


圖 12. Split 時上方追加編碼 0，下方追加編碼 1

| C3  | Symbol        | code |
|-----|---------------|------|
| 0.4 | A2            | 1    |
| 0.3 | A6            | 00   |
| 0.3 | {A4,A3,A5,A1} | 01   |

| C4  | Symbol           | code |
|-----|------------------|------|
| 0.6 | {A6,A4,A3,A5,A1} | 0    |
| 0.4 | A2               | 1    |

圖 13. C4 Split 至 C3

| Sym. | code  |
|------|-------|
| A2   | 1     |
| A6   | 00    |
| A1   | 011   |
| A4   | 0100  |
| A3   | 01010 |
| A5   | 01011 |

| C1  | S.  | code |
|-----|-----|------|
| 0.4 | A2  | 1    |
| 0.3 | A6  | 00   |
| 0.1 | A1  | 011  |
| 0.1 | A4  | 0100 |
| 0.1 | {*} | 0101 |

| C2  | S.  | code |
|-----|-----|------|
| 0.4 | A2  | 1    |
| 0.3 | A6  | 00   |
| 0.2 | {*} | 010  |
| 0.1 | A1  | 011  |

| C3  | S.  | code |
|-----|-----|------|
| 0.4 | A2  | 1    |
| 0.3 | A6  | 00   |
| 0.3 | {*} | 01   |

| C4  | S.  | code |
|-----|-----|------|
| 0.6 | {*} | 0    |
| 0.4 | A2  | 1    |

圖 14. 完整的 Split 過程，圖中{\*}代表多項 symbol 組成的混合集團

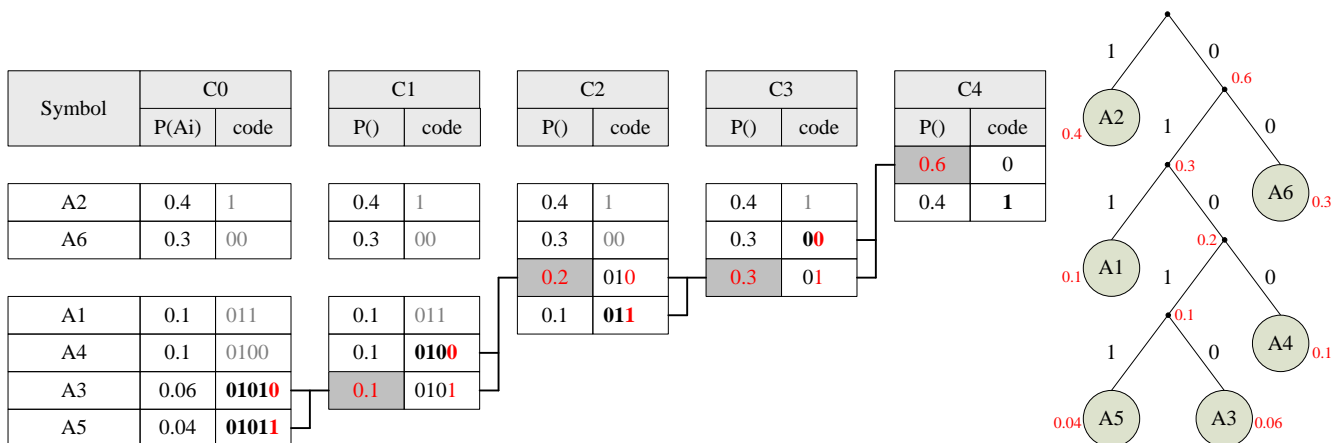


圖 15. 以 Huffman table 來進行 split



故經過 split 階段後，即可得到的例子的 Huffman coding 如下圖所示

| Symbol | P(A <sub>i</sub> ) | Huffman code |
|--------|--------------------|--------------|
| A1     | 0.1                | <b>011</b>   |
| A2     | 0.4                | <b>1</b>     |
| A3     | 0.06               | <b>01010</b> |
| A4     | 0.1                | <b>0100</b>  |
| A5     | 0.04               | <b>01011</b> |
| A6     | 0.3                | <b>00</b>    |

圖 16. 本範例各 symbol 對應的 Huffman code

Huffman coding 是一種 prefix-free 的技術，意即編碼中的任意一個 code 均不是其他編碼的 prefix，因此不需要有間隔(space)的編碼也能做到可變長度編碼的特性。故對圖 17 中的二進位位元流，以圖 16 中的 Huffman code 進行解碼，即代表 A2A4A2A6A1A2A2A6A2 的原文。

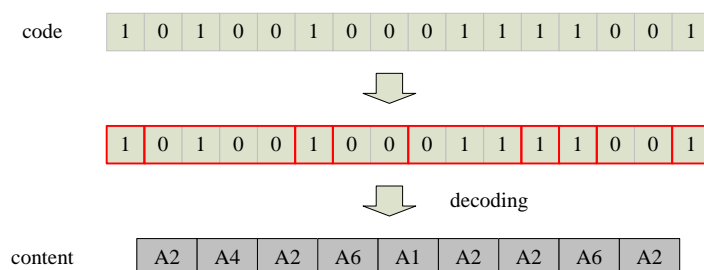


圖 17. 利用 Huffman coding 進行解碼

### 2.3.1 huffman 電路輸出 huffman code 的表示方式

Huffman 電路產生各 symbol 所對應的 Huffman code 時，由於顯示編碼內容的 HC1~HC6 輸出均為固定 8-bit 寬度，而 Huffman code 是可變長度編碼，故額外加上遮罩訊號，以指出 HC1~HC6 輸出的那幾個 bit 是有效的。遮罩中 1 代表有效，0 代表無效。同時 HC1~HC6 輸出超過 Huffman code 編碼長度的位元規定均設為數值 0。

以 A6 的 Huffman code 是 00 為例，該編碼是 2 個 bit 長，所以其遮罩 M6 為 0000\_0011，同時其 HC6 輸出為 0000\_0000。

再以 A3 的 Huffman code 是 01010 為例，該編碼是 5 個 bit 長，所以其遮罩 M3 為 0001\_1111，同時其 HC6 輸出為 0000\_1010。

若有興趣參考原論文，可自行 Google “A Method for the Construction of Minimum-Redundancy Codes, 1952, David A. Huffman”

由統計至產生 Huffman code 這段過程禁止用窮舉查表法，否則以 0 分計算。



## 2.4 時序規格圖

系統輸入/輸出時序規格圖及參數，分別如圖 18 及圖 19 所示。

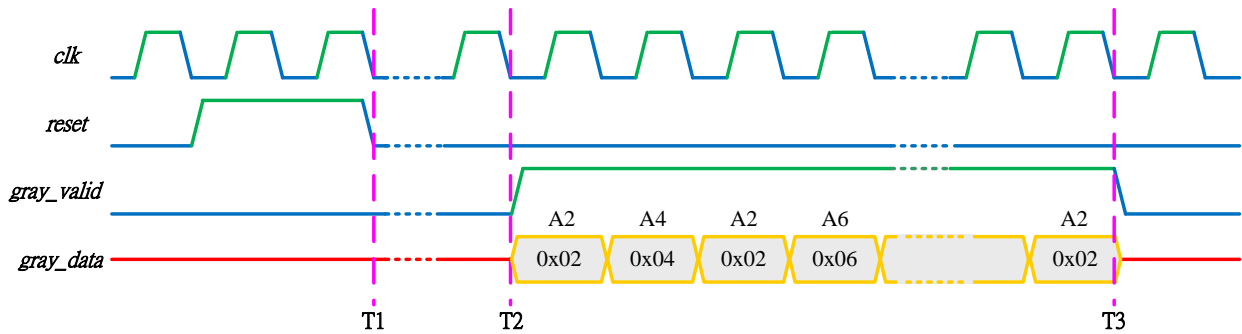


圖 18. 時序規格圖一(資料輸入階段)

- a、T1 時間點，*reset* 訊號持續兩個 Cycle 時間後，huffman 電路初始化結束。
- b、T2 時間點，testbench 端開始傳送灰階圖像資料，合計 100 個 pixel，在時脈訊號負緣觸發一個 cycle 傳送一筆 pixel 的灰階值。資料傳送期間 *gray\_valid* 拉為 High。參賽者所設計之 huffman 電路應採正緣觸發動作來進行設計。
- c、T3 時間點，testbench 傳送資料完畢，將 *gray\_valid* 拉為 Low。此後參賽者所設計之 huffman 電路應接著完成各 symbol 數目的計數，以及推算各 symbol 對應的 Huffman code，此部分請參考圖 19 與輸出相關的相關說明。

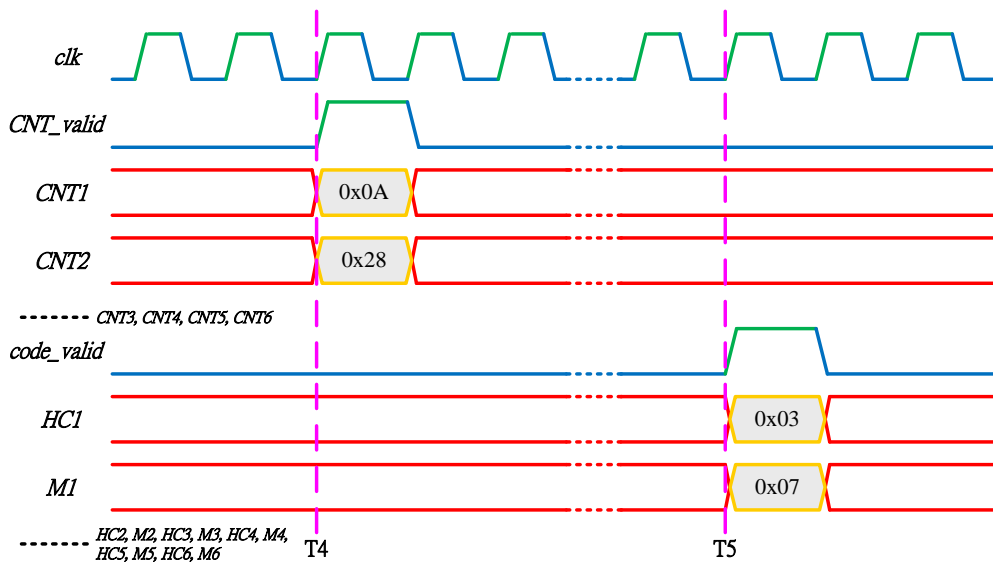


圖 19. 時序規格圖二 (輸出部分)

- d、T4 時間點(參考圖 19)，當 huffman 電路完成各 symbol 數目的計算，應將 *CNT\_valid* 訊號拉為 High 一個時脈周期，並在該周期時間內利用 *CNT1*、*CNT2*、*CNT3*、*CNT4*、*CNT5*、*CNT6* 輸出各 symbol 數目。(注意，huffman 電路為正緣觸發動作)
- e、T5 時間點(參考圖 19)，當 huffman 電路推算出各 symbol 的 Huffman code，應將 *code\_valid* 訊號拉為 High 一個時脈周期，並在該周期時間內利用 *HC1*、*M1*、*HC2*、*M2*、*HC3*、*M3*、*HC4*、*M4*、*HC5*、*M5*、*HC6*、*M6* 輸出各 symbol 的 Huffman code，testbench 偵測到 *code\_valid* 拉回

為 Low 時即可進行驗證，驗證完成後整個模擬會立即結束。(注意，huffman 電路為正緣觸發動作)

### 3. 評分標準

主辦單位的評分人員將依照參賽者提供之系統時脈進行 RTL simulation 或 gate-level simulation，以驗證設計正確性，並且依據設計檔上傳至 CIC FTP 檔案伺服器(請參閱附錄 D)的時間來進行排名。各參賽隊伍應於參賽者定義的系統時脈下，確保輸出結果無設置與保持時間(setup/hold time)的問題，並完全符合主辦單位所提供的標準設計結果。

CIC 將本試題分為下面五個等級作為功能完成度之評分，完成度越高者優先錄取。若為同一等級則以檔案上傳時間來評分，請在每完成新等級時即將結果上傳，CIC 會以最後上傳版本作為評分結果。

1. A 等級：通過所有 3 組測試樣本(包含 CIC 評分用樣本)之 RTL 與 gate-level simulation
2. B 等級：通過其中 3 組測試樣本(包含 CIC 評分用樣本)之 RTL simulation
3. C 等級：通過其中 2 組測試樣本(包含 CIC 評分用樣本)之 RTL simulation
4. D 等級：通過其中 1 組測試樣本(包含 CIC 評分用樣本)之 RTL simulation
5. E 等級：即排除以上狀況者，歸於 E 等級

## 附錄

在附錄 A 中說明本次競賽之軟體環境；附錄 B 為主辦單位所提供各參賽者的設計檔說明；附錄 C 為評分用檔案，亦即參賽者必須回傳至 CIC 的檔案資料；附錄 D 則為設計檔上傳步驟說明；附錄 E 則為測試樣本說明；附錄 F 則為 Huffman Tree。

### 附錄 A 軟體環境

競賽所提供的設計軟體與版本如下表二。驗證評分時，係以所列軟體及版本作為驗證依據。

表二、設計軟體版本

| Functionality         | Corresponding EDA tools  |
|-----------------------|--|
| Logic Simulator       | NC-Verilog (INCISIVE_15.20.039)<br>Modelsim (v10.5c)<br>QuestaSim (v10.5c)<br>VCS-MX (2017.03) |
| Logic Synthesizer     | Design-Compiler (2017.09-SP2)  |
| HDL Debugger          | Verdi (2016.06-sp1-1)<br>nLint (2016.06-sp1-1)   |
| Standard cell library | Cell-Based Design Kit for IC Contest v2.1  |

## 附錄 B 設計檔案說明

### 1. 下表三.為主辦單位所提供各參賽者的設計檔案

表三、設計檔

| 檔名                       | 說明                             |
|--------------------------|--------------------------------|
| tb.v                     | 測試樣本檔。此測試樣本檔定義了時脈週期與測試樣本之輸入信號。 |
| huffman.v ( huffman.vhd) | 參賽者所使用的設計檔，已包含系統輸/出入埠之宣告       |
| ./pattern1.dat           | 第一組測試樣本檔案                      |
| ./pattern2.dat           | 第二組測試樣本檔案                      |
| ./pattern3.dat           | 第三組測試樣本檔案                      |
| ./golden1.dat            | 第一組比對樣本檔案                      |
| ./golden2.dat            | 第二組比對樣本檔案                      |
| ./golden3.dat            | 第三組比對樣本檔案                      |
| report.000               | 結果報告範本                         |
| huffman.sdc              | Design Compiler 電路合成規範檔        |
| DC_syn.tcl               | Design Compiler 合成之參考 script   |
| synopsys_dc.setup        | Design Compiler 初始設定範例檔案       |

### 2. 請使用 huffman.v (.vhd)，進行本題電路之設計。其 Verilog 模組名稱、輸出/入埠宣告如下所示：

```
`timescale 1ns/10ps
module huffman ( clk, reset, gray_valid, gray_data, CNT_valid, CNT1, CNT2, CNT3, CNT4, CNT5, CNT6,
                code_valid, HC1, HC2, HC3, HC4, HC5, HC6, M1, M2, M3, M4, M5, M6);
    input      clk;
    input      reset;
    input      gray_valid;
    input  [7:0] gray_data;
    output      CNT_valid;
    output  [7:0] CNT1, CNT2, CNT3, CNT4, CNT5, CNT6;
    output      code_valid;
    output  [7:0] HC1, HC2, HC3, HC4, HC5, HC6;
    output  [7:0] M1, M2, M3, M4, M5, M6;
    //=====

    //=====
Endmodule
```

3. 比賽共提供三組測試樣本，參賽者可依下面範例來進行模擬：  
(此處說明以 ncverilog 為主，若使用其他 simulator 軟體請自行參閱相關使用手冊)

RTL Simulation 時使用指令如下：

- 使用 ncverilog 模擬指令範例如下：

```
ncverilog tb.v huffman.v +define+tb1 +access+r
```

- 使用 modelsim 模擬，則是在 compiler verilog 時，使用下面指令：

```
vlog tb.v huffman.v +define+tb1 +access+r
```

Gate-Level Simulation 時使用指令如下：

- 使用 ncverilog 模擬指令範例如下：

```
ncverilog tb.v huffman_syn.v -v tsmc13_neg.v +define+SDF +define+tb1 \  
+access+r
```

define 中加上 SDF 可讓測試程式引入 gate level netlist 的 sdf 檔案資訊。

若要避免時序檢查以減少錯誤訊息，可於模擬指令中自行加入+notimingchecks。

- 上述指令中+define+tb1 指的是使用第一組測試樣本模擬，當使用第二組測試樣本請自行修改此參數為+define+tb2；當使用第三組測試樣本，請自行修改此參數為+define+tb3。
- tsmc13\_neg.v 為 gate-level simulation 時所使用的 standard cell simulation model，該檔案位於 CBDK\_IC\_Contest\_v2.1 裡(參考路徑為 CBDK\_IC\_Contest\_v2.1/Verilog/tsmc13\_neg.v)。

## 附錄 C 評分用檔案

評分所需檔案可分為三部份：(1)RTL design，即各參賽隊伍對該次競賽設計的 RTL code，若設計採模組化而有多個設計檔，請務必將合成所要用的各 module 檔放進來，以免評審進行評分時，無法進行編譯；(2)gate-level design，即由合成軟體所產生的 gate-level netlist，以及對應的 SDF 檔；(3)report file，參賽隊伍必須依照自己的設計內容，撰寫 report.000 檔，以方便主辦單位進行評分，report.000 的格式如圖八所示。(report 檔以後三碼序號表示版本，若繳交檔案更新版本，則新版的 report 檔檔名為 report.001，依此類推)

表六、評分用檔案

| RTL category                           |              |  |
|--|--------------|--|
| Design Stage                           | File         | Description  |
| N/A                                    | report.xxx   | design report  |
| RTL Simulation                         | *.v or *.vhd | Verilog (or VHDL) synthesizable RTL code                         |
| Gate-Level category                    |              |  |
| Design Stage                           | File         | Description  |
| Pre-layout<br>Gate-level<br>Simulation | *_syn.v      | Verilog gate-level netlist generated by Synopsys Design Compiler |
|  | *_syn.sdf    | SDF timing information generated by Synopsys Design Compiler     |
|  | *_syn.ddc    | design database generated by Synopsys Design Compiler            |

FTP 帳號(FTP account): 999999

通過 gate-level simulation 之 cell area report : 9000

通過 gate-level simulation 之 clock cycle time (ns) : 10

--- RTL category---

使用之 HDL 模擬器名稱(HDL simulator ): nc-verilog

RTL 檔案名稱(RTL filename): huffman.v 以及使用到的子模組檔案...

--- Pre-layout gate-level ---

gate-level 檔案名稱(gate\_level filename): huffman\_syn.v

gate-level sdf filename: huffman\_syn.sdf

design compiler 合成資料庫(dc library): huffman\_syn.ddc

-----注意事項(annotation)-----

(其餘注意事項依各參賽隊伍的需求填寫)

圖八、report.000 的範本



## 附錄 D 檔案上傳

所有包含於如附錄 C 中表格所示的檔案，均需要提交至 CIC。並且，提交的設計檔案，需要經過壓縮於同一個資料夾下，步驟如下：

1. 建立一個 result\_xxx 資料夾。其中“xxx”表示繳交版本。例如“000”表示為第一次上傳；“001”表示為第二度上傳；002 表示為第三度上傳，以此類推…。
2. 參考附錄 C 評分用檔案，將所有繳交檔案複製到 result\_xxx 資料夾
3. 執行 tar 指令將 result\_xxx 資料夾包裝起來，tar 的指令範例如下：  
tar cvf result\_xxx.tar result\_xxx  
其中 xxx 表示繳交版本  
執行完後應該會得到 result\_xxx.tar 的檔案
4. 使用 ftp 將 result\_xxx.tar 及 report.xxx 一併上傳至 CIC 提供的 ftp server，result\_xxx.tar 與 report.xxx 之“xxx”編號需一致，評審將以最後上傳的設計檔及報告檔編號進行評分作業。

本題限制上傳之設計檔僅可使用 tar 或 zip 壓縮格式，使用 rar 或其他格式者一律不予計分。

請注意!!上傳之 FTP 需切換為二進制模式(binary mode)，且傳輸埠均設為 21(port:21)。

ftp 的帳號和密碼在賽前已用 email 寄給各參賽者。若有任何問題，請聯絡 CIC

FTP site1 (新竹晶片中心)：iccftp.cic.org.tw (140.126.24.18)

FTP site2 (南區晶片中心)：iccftp2.cic.org.tw(140.110.117.9)

5. 若你需要繳交更新版本，請重覆以上步驟，並記得修改 report 檔及 tar 檔的版本編號，因為你無法修改或刪除或覆蓋之前上傳的資料。

附錄 E 測試樣本

在此提供參賽者兩份測試樣本，以供驗證所設計之電路是否合乎標準。CIC 於評分階段會用額外的測試樣本測試參賽者所設計之電路。**禁止用窮舉查表法，否則以 0 分計算。**

測試樣本 1

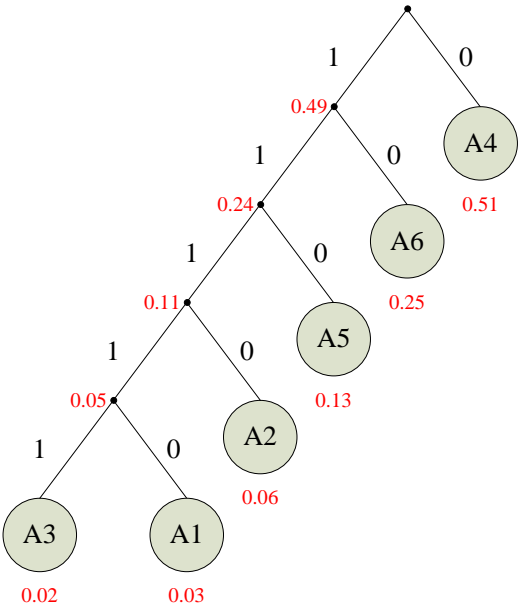
Pattern file : pattern1.dat

Golden file : golden1.dat

對應之 Huffman table 如下所示

| Symbol | C0    |       | C1   | C2   | C3   | C4   |
|--------|-------|-------|------|------|------|------|
|        | P(Ai) | Code  | P()  | Code | P()  | Code |
| A4     | 0.51  | 0     | 0.51 | 0    | 0.51 | 0    |
| A6     | 0.25  | 10    | 0.25 | 10   | 0.25 | 10   |
| A5     | 0.13  | 110   | 0.13 | 110  | 0.24 | 11   |
| A2     | 0.06  | 1110  | 0.06 | 1110 | 0.11 | 111  |
| A1     | 0.03  | 11110 | 0.05 | 1111 |      |      |
| A3     | 0.02  | 11111 |      |      |      |      |

| Symbol | P(Ai) | Huffman code | CNTi      | HCi                  | Mi                  |
|--------|-------|--------------|-----------|----------------------|---------------------|
| A1     | 0.03  | 11110        | 3 = 0x3   | 8'b00001_1110 = 0x1E | 8'b0001_1111 = 0x1F |
| A2     | 0.06  | 1110         | 6 = 0x6   | 8'b0000_1110 = 0x0E  | 8'b0000_1111 = 0x0F |
| A3     | 0.02  | 11111        | 2 = 0x2   | 8'b00001_1111 = 0x1F | 8'b0001_1111 = 0x1F |
| A4     | 0.51  | 0            | 51 = 0x33 | 8'b0000_0000 = 0x00  | 8'b0000_0001 = 0x01 |
| A5     | 0.13  | 110          | 13 = 0x0D | 8'b0000_0110 = 0x06  | 8'b0000_0111 = 0x07 |
| A6     | 0.25  | 10           | 25 = 0x19 | 8'b0000_0010 = 0x02  | 8'b0000_0011 = 0x03 |



測試樣本 2

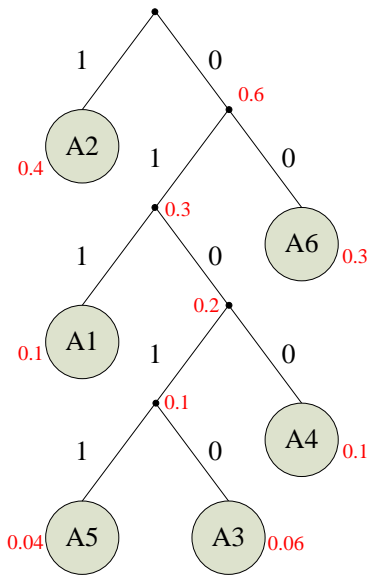
Pattern file : pattern2.dat

Golden file : golden2.dat

對應之 Huffman table 如下(同圖 15)所示

| Symbol | C0    |       | C1  |      | C2  |      | C3  |      | C4  |      |
|--------|-------|-------|-----|------|-----|------|-----|------|-----|------|
|        | P(Ai) | code  | P() | code | P() | code | P() | code | P() | code |
| A2     | 0.4   | 1     | 0.4 | 1    | 0.4 | 1    | 0.4 | 1    | 0.6 | 0    |
| A6     | 0.3   | 00    | 0.3 | 00   | 0.3 | 00   | 0.3 | 00   | 0.4 | 1    |
| A1     | 0.1   | 011   | 0.1 | 011  | 0.2 | 010  | 0.3 | 01   |     |      |
| A4     | 0.1   | 0100  | 0.1 | 0100 | 0.1 | 011  |     |      |     |      |
| A3     | 0.06  | 01010 | 0.1 | 0100 |     |      |     |      |     |      |
| A5     | 0.04  | 01011 | 0.1 | 0101 |     |      |     |      |     |      |

| Symbol | P(Ai) | Huffman code | CNTi      | HCi                 | Mi                  |
|--------|-------|--------------|-----------|---------------------|---------------------|
| A1     | 0.1   | 011          | 10 = 0xA  | 8'b0000_0011 = 0x03 | 8'b0000_0111 = 0x07 |
| A2     | 0.4   | 1            | 40 = 0x28 | 8'b0000_0001 = 0x01 | 8'b0000_0001 = 0x01 |
| A3     | 0.06  | 01010        | 6 = 0x6   | 8'b0000_1010 = 0x0A | 8'b0001_1111 = 0x1F |
| A4     | 0.1   | 0100         | 10 = 0xA  | 8'b0000_0100 = 0x04 | 8'b0000_1111 = 0x0F |
| A5     | 0.04  | 01011        | 4 = 0x4   | 8'b0000_1011 = 0x0B | 8'b0001_1111 = 0x1F |
| A6     | 0.3   | 00           | 30 = 0x1E | 8'b0000_0000 = 0x00 | 8'b0000_0011 = 0x03 |

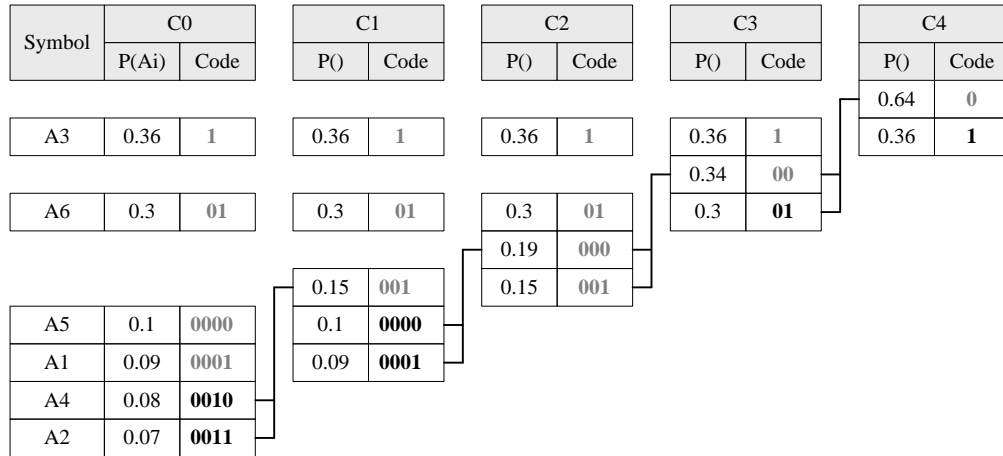


### 測試樣本 3

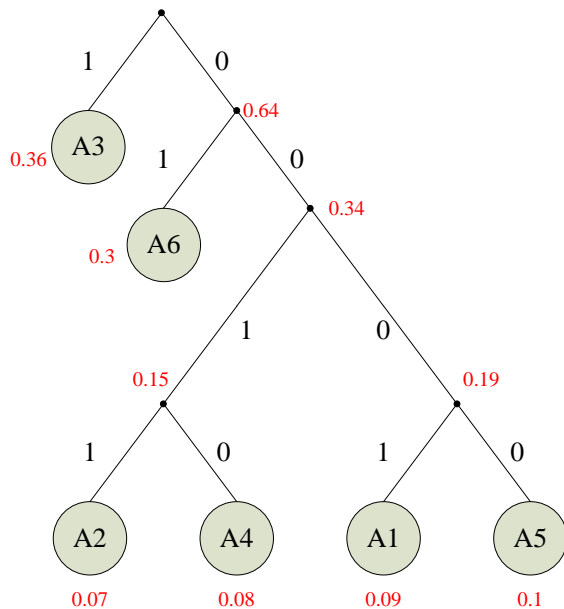
Pattern file : pattern3.dat

Golden file : golden3.dat

對應之 Huffman table 如下所示



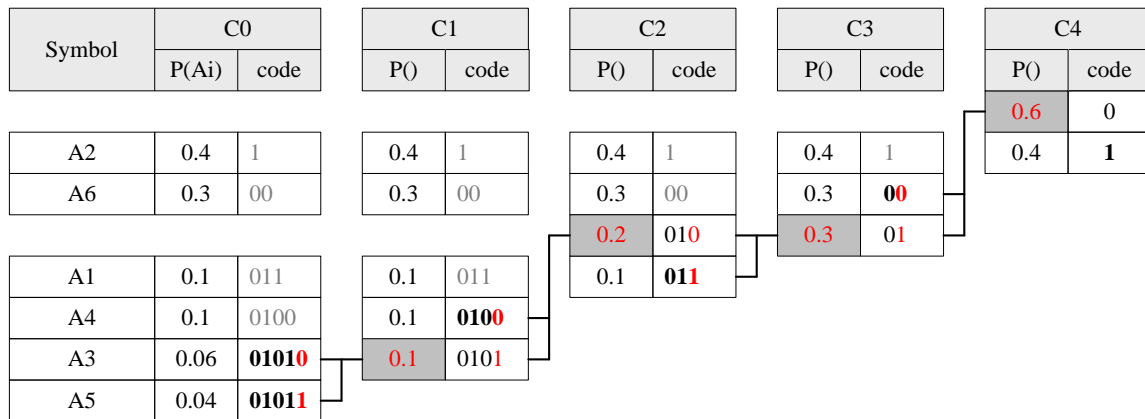
| Symbol | P(Ai) | Huffman code | CNTi      | HCi                 | Mi                  |
|--------|-------|--------------|-----------|---------------------|---------------------|
| A1     | 0.09  | 0001         | 9 = 0x9   | 8'b0000_0001 = 0x01 | 8'b0000_1111 = 0x0F |
| A2     | 0.07  | 0011         | 7 = 0x7   | 8'b0000_0011 = 0x03 | 8'b0000_1111 = 0x0F |
| A3     | 0.36  | 1            | 36 = 0x24 | 8'b0000_0001 = 0x01 | 8'b0000_0001 = 0x01 |
| A4     | 0.08  | 0010         | 8 = 0x8   | 8'b0000_0010 = 0x02 | 8'b0000_1111 = 0x0F |
| A5     | 0.1   | 0000         | 10 = 0xA  | 8'b0000_0000 = 0x00 | 8'b0000_1111 = 0x0F |
| A6     | 0.3   | 01           | 30 = 0x1E | 8'b0000_0001 = 0x01 | 8'b0000_0011 = 0x03 |



## 附錄 F Huffman Tree

為幫助參賽者對 Huffman coding 的了解，在此提供 Huffman Tree 畫法供各位參考。此處直接使用題目本文中的範例(即測試樣本 pattern2.dat)做說明。

測試樣本 2 的 Huffman table 如下所示



要建 Huffman tree 時，首先還是由樣本資料統計後的各 symbol 出現機率的排序後結果出發。測試樣本的初始排序資料如下(由上至下對應到由大到小，最底下 2 個是機率最小的，在作答期間，雖題目中多處用到機率的數值去比大小，但實際上可以直接用計數出來的出現次數代替)。在初始化排序時，對相同機率的 symbol 何者在上會影響做出來的 Huffman coding，雖然壓縮率的表現一樣，但此處為了評分方便，統一規定在初始排率時相同機率者，擁有較小索引值的 symbol 排上方，所以下圖中，A1 與 A4 的機率是相同的，但 A1 的索引較小，故 A1 排 A4 上方。

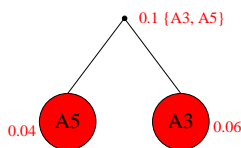
| C0     |       |
|--------|-------|
| Symbol | P(Ai) |
| A2     | 0.4   |
| A6     | 0.3   |
| A1     | 0.1   |
| A4     | 0.1   |
| A3     | 0.06  |
| A5     | 0.04  |

接著會經過四個回合，每個回合會合併最下面兩個(即當時機率最小的兩組)成一個合併項，再與原來的資料作排序。此處合併項{A3,A5}的機率是  $0.06+0.04=0.1$ ，再與上面四個項的機率作排序，發現跟 0.1 機率的 A1 與 A4 相同機率，在此合併的過程中，同上面理由，統一規定合併過程中合併項若有相同機率的現象，合併項放在此相同機率的群組中的最後一個。所以合併後{A3,A5}放在 A4 的下面。此時對應的局部 Huffman tree 如下圖(Huffman 的精神在於每次找最低機率的兩個來畫圖)，圖中代表由 A3 與 A5 的合併項，統一規定左邊的機率要小於右邊的，故 A3 放右邊。

| C0                 |        |
|--------------------|--------|
| P(A <sub>i</sub> ) | Symbol |
| 0.4                | A2     |
| 0.3                | A6     |
| 0.1                | A1     |
| 0.1                | A4     |
| 0.06               | A3     |
| 0.04               | A5     |

| P(A <sub>i</sub> ) | Symbol   |
|--------------------|----------|
| 0.1                | {A3, A5} |



該圖表格合併後作排序的結果如下

| C0                 |        |
|--------------------|--------|
| P(A <sub>i</sub> ) | Symbol |
| 0.4                | A2     |
| 0.3                | A6     |
| 0.1                | A1     |
| 0.1                | A4     |
| 0.06               | A3     |
| 0.04               | A5     |

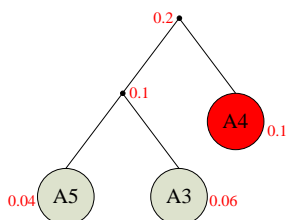
| C1                 |          |
|--------------------|----------|
| P(A <sub>i</sub> ) | Symbol   |
| 0.4                | A2       |
| 0.3                | A6       |
| 0.1                | A1       |
| 0.1                | A4       |
| 0.1                | {A3, A5} |

接著我們利用 C1 排序後的表格進行下一次合併，找 C1 表格中最低機率的兩個來合併(畫到 Huffman tree 內)，若有機率相同者，取最底下兩個作合併，所以此處為 A4 跟 {A3, A5} 作合併，合併的兩者在表格上方(機率大的)的放右邊，所以得到此時對應的 Huffman tree 如下

| C1                 |          |
|--------------------|----------|
| P(A <sub>i</sub> ) | Symbol   |
| 0.4                | A2       |
| 0.3                | A6       |
| 0.1                | A1       |
| 0.1                | A4       |
| 0.1                | {A3, A5} |

| P(A <sub>i</sub> ) | Symbol       |
|--------------------|--------------|
| 0.2                | {A4, A3, A5} |



該圖表格合併後作排序的結果如下，因為合併項 {A4, A3, A5} 的機率 0.2 比 A1 機率大，所以在表格中把 A1 擠下去了

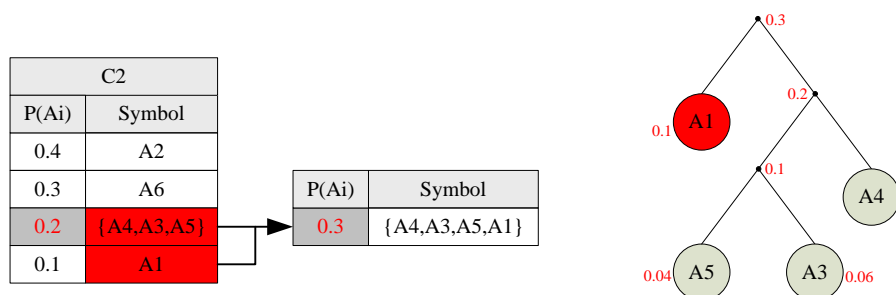
| C1                 |          |
|--------------------|----------|
| P(A <sub>i</sub> ) | Symbol   |
| 0.4                | A2       |
| 0.3                | A6       |
| 0.1                | A1       |
| 0.1                | A4       |
| 0.1                | {A3, A5} |

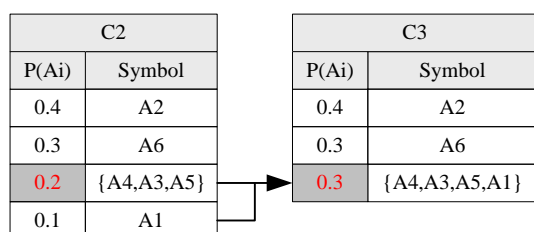
| C2                 |              |
|--------------------|--------------|
| P(A <sub>i</sub> ) | Symbol       |
| 0.4                | A2           |
| 0.3                | A6           |
| 0.2                | {A4, A3, A5} |
| 0.1                | A1           |

接著我們利用 C2 排序後的表格進行下一次合併，找 C2 表格中最低機率的兩個來合併(畫到

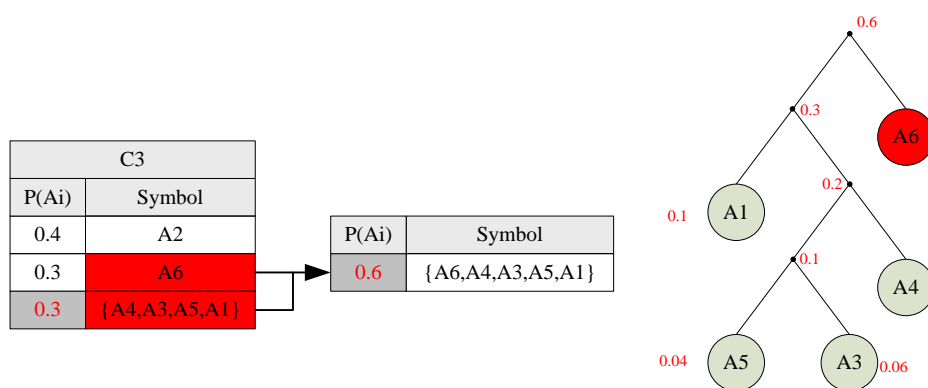
Huffman tree 內)，此時原合併項的機率比較大放右邊，新合併的 A1 機率比較小，放左邊



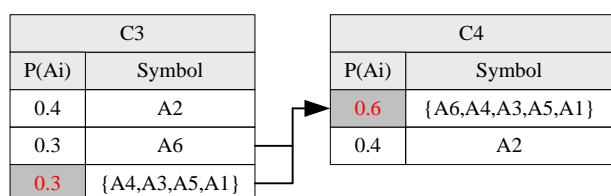
該圖表格合併後作排序的結果如下，因為合併項{A1,A4,A3,A5}的機率 0.3 與 A6 機率一樣大，所以依規定放最下面(合併過程中合併項若有相同機率的現象，合併項放在此相同機率的群組中的最後一個)



接著我們利用 C3 排序後的表格進行下一次合併，找 C3 表格中最低機率的兩個來合併(畫到 Huffman tree 內)，此時原合併項的機率比較小大放左邊，新合併的 A6 機率比較大，放右邊



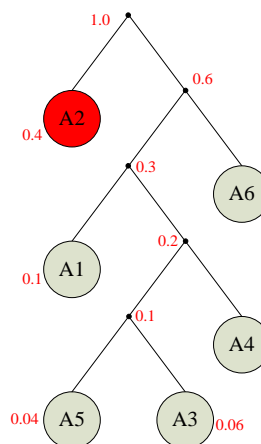
該圖表格合併後作排序的結果如下，因為合併項{A6,A1,A4,A3,A5}的機率 0.6 大於 A2 的機率，所以在表格中把 A2 擠下去了。



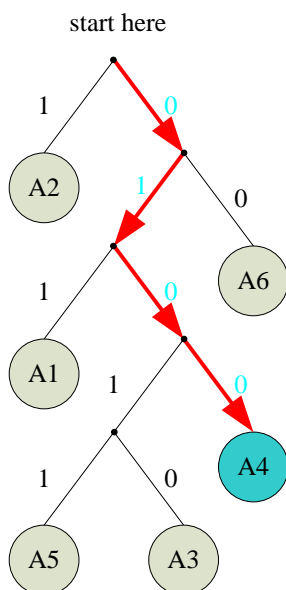


至此 C4 回合，表格只剩下兩個項，繪 Huffman tree 時需作最後一次合併(合起來總機率為 1) ，此時的 Huffman tree 如下所示，此時原合併項的機率比較大放大右邊，新合併的 A2 機率比較小，放左邊。

| C4    |                  | C5    |                     |
|-------|------------------|-------|---------------------|
| P(Ai) | Symbol           | P(Ai) | Symbol              |
| 0.6   | {A6,A4,A3,A5,A1} | 1.0   | {A2,A6,A4,A3,A5,A1} |
| 0.4   | A2               |       |                     |



如何由 Huffman tree 得到個 symbol 的 Huffman coding 呢？至此對 Huffman tree 的每一組分支，左邊設為 1，右邊設為 0，由頂端沿 Huffman tree 走到各 symbol 沿途經過的數字即為各 symbol 的 Huffman code。如下圖以 A4 的 Huffman code，即為由 Top (start here)開始走到 A4 沿途遇到的數字，所以 A4 的 Huffman code 為 0100，與用 Huffman table 方法得到的結果一致。



| Symbol | P(Ai) | Huffman code |
|--------|-------|--------------|
| A1     | 0.1   | <b>011</b>   |
| A2     | 0.4   | <b>1</b>     |
| A3     | 0.06  | <b>01010</b> |
| A4     | 0.1   | <b>0100</b>  |
| A5     | 0.04  | <b>01011</b> |
| A6     | 0.3   | <b>00</b>    |