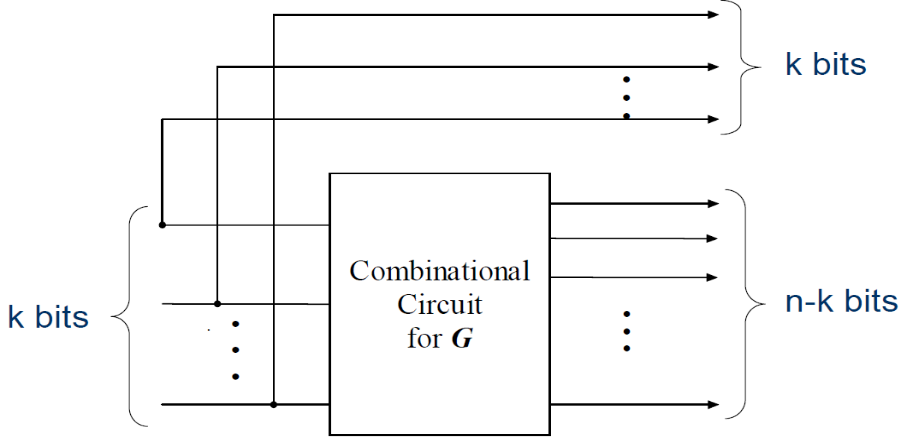
**實習單元一、Hamming code**

1. **原理介紹**

Systematic Encoder for General (n，k) Codes

****

以數學型式表示如圖1.2作為範例，假設I=()=(1101)，C=()=(1101010)經過運算在關係示表式則為:

　　　　C=I，C=I，C=I，C=I，

C= I+I+I，C= I+I+I，C= I+I+I

在下面的例子做說明，假設有三個集合(sets)Ａ、Ｂ、Ｃ兩兩相交（如圖1.1），要編碼碼初4位元的編碼器採用3個額外(冗餘)的位元由偶校正來檢驗。A集合由編號1、2、3、5表示；B集合由編號1、2、4、6表示；C集合由編號1、3、4、7表示。

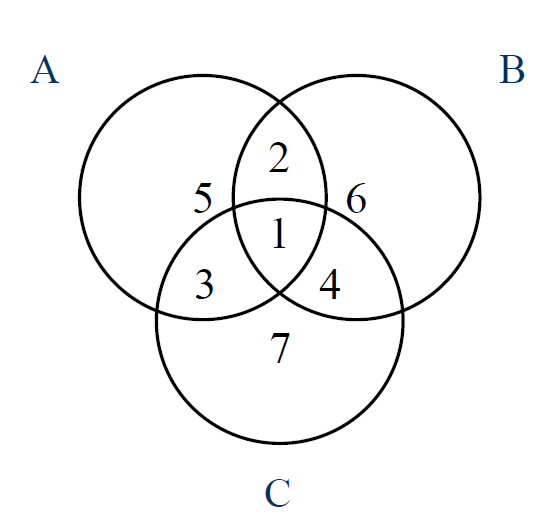


　　　 圖1.1

我們把每一小塊編號為１,2,3,,7，然後假設原來傳1101，依序把它寫在編號1,2,3,4的空格上，然後多加三個bits上去，讓每個圓圈裡面1的個數是even parity，就如圖1.2。

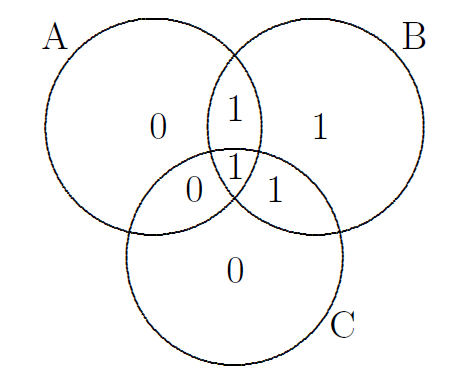


圖1.2

所以加進去的三個bits是010，最後要傳的是1101010，要多加一些東西讓他們有數學結構，這樣就可以更正錯誤。

　　在下面的例子，如果原本傳的是1101010，假設第6個bit錯了，1變成0，如圖1.3 a。

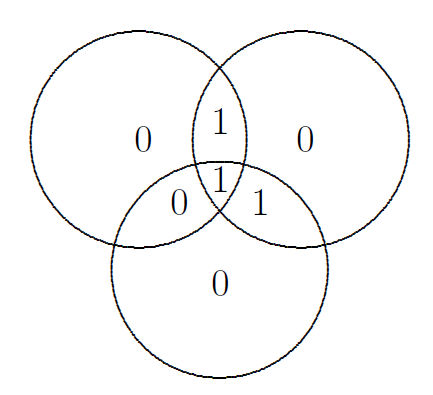


圖1.3a

解碼器有個數學結構，就是每個圓裡面1的個數是偶數，所以A有兩個1，沒有問題，C有兩個1，也沒有問題，可是B有3個1，有問題，因此我們知道是6這一塊錯了，就把這個0改回來變成1，以符合原本的結構。再舉一個例子，假如錯第3個bit，其他都對，如圖1.3b，那麼A有3個1，有問題，B有4個1，沒有問題，C有3個1，也有問題，所以我們就找這一塊，也就是3這一塊錯了，就把它改回來。同樣道理，如果是錯在第一個bit，如圖1.3c，三個圓都有問題，所以你知道是錯在這一塊，然後把它改回來變成1。所以這個碼可以改一個錯，我們就成做single-error correcting(SEC)。

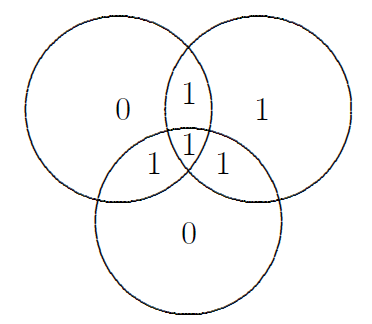


圖1.3b

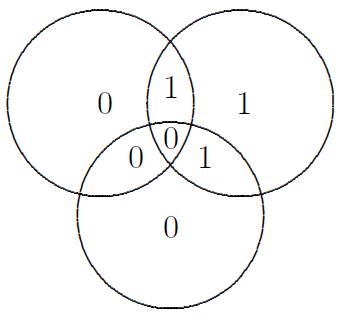


圖1.3c

這個碼有一個術語叫做(7，4) Hamming code，因為全部長度是7，而只有4個bits是原來要傳的資料，所以叫(7，4)。

　　假如有兩個錯會怎麼樣？比方說，如果是6、7兩個bits 錯了，如圖1.4。那麼A不會有問題，B 和C 都有問題，照剛才的方法就要改4這個bit，結果除了原來的兩個錯，反而又多出了一個錯，事實上，我們可以證明任何兩個bits錯，如果照原來的方法去改，都會再多出一個錯，為什麼？因為這個碼不能改兩個錯， 最多只能改一個錯，這跟我們加了多少個多餘的bits 有關，在這個例子我們多加了3個bits，如果要改兩個或兩個以上的錯，就要多加新的bits。

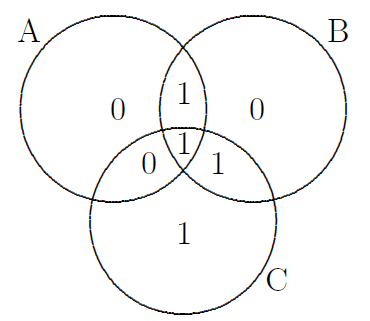


圖1.4

1. **程式碼**

這次的作業我使用的軟體是C語言

|  |  |
| --- | --- |
| 圖2.1 | 圖2.1 程式解釋  一開始先輸入4個數字x0、x1、x2、x3，接著利用XOR運算，開始計算x4、x5、x6的值，即可得到編碼後的(7,4) code。  再來輸入我們接收到的值，稱為y，此時系統會開始運算y值是否符合XOR運算，並且另外計算一個e值。  Ps.輸入接收值時，只能錯誤一個位元，否則會無法除錯。 |
| 圖2.2 | 圖2.2 程式解釋  如果e值等於0，代表我輸入的接收值是正確的，就會出現No error的情況；如果e值不等於0，就代表有位元出現錯誤，系統就會計算出正確的A、B、C值並顯示出來，最後再顯示正確的接收值。  如果錯誤位為0，就對它進行補數，反之亦然 |

1. **執行結果**

當我輸入圖3.1的1、2、3、4為1、0、0、0時，程式會直接執行編碼解果，如圖3.2所示，顯示正確的編碼結果1、0、0、0、1、1、1。

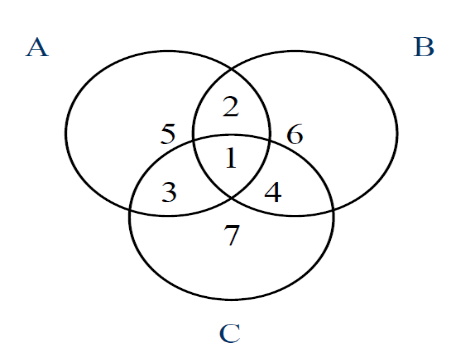
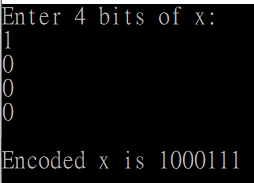
 

圖3.1 圖3.2

當我輸入和編碼結果相同的值當作接收值時，程式出現No error的情況，代表沒有任何位元出現錯誤，如圖3.3所示。

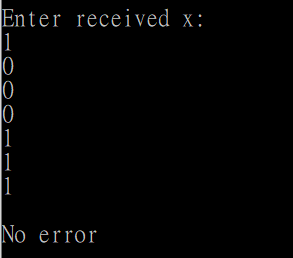


圖3.3

在我已得知正確的Hamming code後，我要來嘗試偵測錯誤的位元：先輸入接收到的錯誤，然後程式會把正確的結果顯示出來，如圖3.4所示。

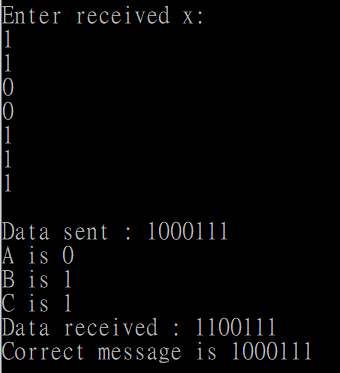


圖3.4

當我輸入2個位元的錯誤時，程式就會無法進行除錯，就無法顯示正確的編碼，如圖3.5所示。

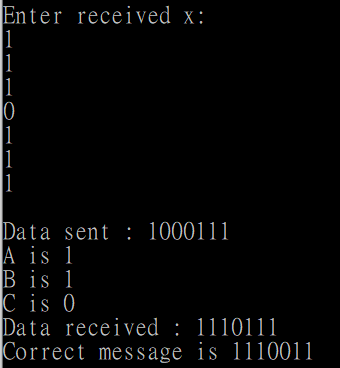


圖3.5