

多媒體通訊期末報告

MPEG 之 AAC 介紹

姓名：李炯文、吳子懿

學號：E24893217、E24894182

班別：電四丙

負責部分：李(一、三、四、六)吳(二、五、七)

一 AAC 簡介

AAC 的全名為 Advanced Audio Coding，其意為高階音訊編碼，是國際標準組織 (ISO)訂的音訊標準格式，也是 MPEG 規格的一部分。取樣頻率選擇性更高，更接近 CD 音質；並採用了分辨率更高的濾波器組，達到很高的壓縮率，並可大幅降低傳輸時間及減少儲存空間，適合新一代音樂產品使用。

AAC 的特點：

提升的壓縮率 ： 可以以更小的檔大小獲得更高的音質

支持多聲道 ： 可提供最多 48 個全音域聲道

更高的解析度 ： 最高支援 96KHz 的取樣頻率

提升的解碼效率 ： 解碼播放所占的資源更少

AAC 是個大家族，目前已經制定了如下的 9 種規格，以適應不同場合的需要：

文件	規格
MPEG-2 AAC LC	低複雜度規格 (Low Complexity)
MPEG-2 AAC Main	主規格
MPEG-2 AAC SSR	可變取樣率規格 (Scalable Sampling Rate)
MPEG-4 AAC LC	低複雜度規格
MPEG-4 AAC Main	主規格

MPEG-4 AAC SSR	可變取樣率規格 (Scalable Sampling Rate)
MPEG-4 AAC LTP	長時期預測規格 (Long Term Prediction,)
MPEG-4 AAC LD	低延遲規格 (Low Delay,)
MPEG-4 AAC HE	高效率規格 (High Efficiency)

(1) **Main (主規格)** 包含了除增益控制 (Gain Control) 之外的全部功能，其音質最好。

(2) **低複雜度規格 (Low Complexity)** 則使用比較簡單的時域雜訊修整 (Temporal Noise Shaping, TNS) 模組，也缺少了預測 (Prediction) 和增益控制 (Gain Control) 模組，以此降低複雜度，提高了編碼效率。SSR (Scalable Sampling Rate, 可變取樣率規格) 和 LC 規格大體相似，但多了增益控制功能。

(3) **MPEG-4 AAC LTP (Long Term Prediction, 長時期預測規格)**、**MPEG-4 AAC LD (Low Delay, 低延遲規格)** 和 **MPEG-4 AAC HE (High Efficiency, 高效率規格)** 都是用在低碼率下編碼，尤其是 MPEG-4 AAC HE 規格由於高效率高音質 (低碼率下而言)，再加上有 Nero AAC 編碼器的支援，所以被越來越多地應用到低碼率編碼種。事實上，Main 規格和 LC 規格的音質相差並不大，但是編碼效率卻相差明顯，所以目前使用地最多的當數 LC 規格。無怪乎很多編碼器甚至只支援 LC 規格了！

和 MP3 一樣，炙手可熱的 AAC 格式也催生了大量的編碼器。包括

商業性質的和完全免費的，技術保密的和源碼開放的，對消費者提供的和對開發者提供的，各種各樣的編碼器都有。其中主要的編碼器有：

(1) **FhG**：Fraunhofer IIS 研發的權威編碼器，擁有很好的音質，可惜現在不對外提供了！

(2) **Nero AAC**：可能是目前最完美的 AAC 編碼器了，同時支援 LC AAC 和 HE AAC 規格。這個編碼器是商業性質的，隨著包含了 Nero Digital 的 Nero 6 一起發佈，可能很多有燒錄機的電腦裏已經安裝了該軟體。Nero AAC 編碼器提供了品質最好的 VBR LC AAC 格式，同時 HE AAC 規格保證了它在低碼率下也有不俗的表現。此外，這個編碼器還支援多聲道編碼！

(3) **QuickTime/iTune**：蘋果公司的兩款多媒體軟體都提供了 AAC 編碼功能，其編碼技術來自杜比實驗室（Dolby Laboratories）。起初，它們的編碼器功能比較簡單，只支援雙聲道的 CBR 模式 LC AAC 格式，不過最近蘋果剛剛在 QuickTime Pro 版中提供了 VBR 編碼。本文後面將介紹的 iTunes 儘管還不具備 VBR 模式編碼，但是它卻是目前音質最好的中碼率編碼器！

(4) **FAAC**：最好的命令行編碼器，只支援 LC 規格。其表現中規中矩，最近幾個月新版本的發佈將其音質提升了一個臺階。這個免費的編碼器已經完全能和商業性質的編碼器相媲美了，是搭配 EAC 的不二選擇。

(5) **Psytel**：這是最早期的命令行編碼器，曾經很是風光，其音質在以前是最好的，就是編碼速度很慢。可惜的是其作者現在為 Nero 工作了，這個編碼器也從此停止了更新。因為有相似的 FAAC，所以不建議使用它。

(6) **Coding Technologies**：該公司的 AAC 編碼技術已經被 Real 公司加入到他們的產品中了。最新的 RealProducer 10 和 RealPlayer 10 都包含了這個 AAC 編碼器。

(7) **HHI/zPlane (Compaact!)** : Compaact! 是一款新出的 AAC 編碼器，和 Nero 一樣，也是商業性質的。它的特性相當誘人，支援 LC 和 Main 規格，支援 CBR 和 VBR 模式，支援多聲道高取樣率（24bit／96KHz），還支援命令行操作！官方網站（www.compaact.com）上有試用版可供下載，不過不註冊的話只能使用 50 次。

(8) **Dolby AAC** : 杜比公司的 AAC 編碼器，因為著重在網路廣播，所以低碼率下優勢明顯。據稱它編碼的 48kbps 身歷聲音質比其他 AAC 編碼器好 20%，甚至 64kbps 的 Dolby AAC 可媲美 128kbps 的 MP3！

本次實作的目的是對 AAC 的架構做介紹並比較其與 MP3、WMA、OGG 之壓縮率。

二 AAC 理論和演算法

當今在音訊壓縮可分為兩大類別，一種是以無失真 (Lossless) 為宗旨，指的是當壓縮過的訊號經過解壓縮程序之後，可以還原成原來未經壓縮的訊號一樣，這種方法對於音樂品質有極嚴苛的要求，然而所付出的代價是壓縮率並不高，像 DVD Audio 規格採用的 MLP (Meridian Lossless Packing) 就是無失真壓縮的一例，壓縮率只有 50% 左右。另一種則相反，為得到較高的壓縮效率，根據人類聽覺特性，將一般人耳無法查覺其變化的音軌捨棄，減少音訊編碼的資料量，與解壓縮後的資料不盡相同也無妨，這一種壓縮方式泛稱為感知編碼 (Perceptual Coding) 或是感觀式編碼，目前已成為數位音訊壓縮方法的主流，如 MPEG 及大家所熟知的杜比 AC3 都是採用感知編碼的方式得到較高的壓縮效率。

MPEG Audio 所採用的感知編碼乃根據人類聽覺系統特性所發展出了的人耳聲學 (Psychoacoustic) 為基礎，在人耳聲學的研究中有許多研究成果在現今大量的被音訊壓縮技術所使用，最重要的兩個應用可以歸納成頻率軸 (Frequency Domain) 與時間軸 (Time Domain) 的遮罩效應[1]：

(1) 頻率軸上的遮罩性：當有某一單頻音出現時，會影響到人耳對於臨近頻率聽覺的靈敏度，如圖 1 中，當 1kHz 單頻音的音量大於鄰近頻率的音量時，臨近的頻率如 980、1020Hz 的聲音就會被遮蓋住，感受不到這兩個頻率的聲音，此稱為頻率上的遮罩效應

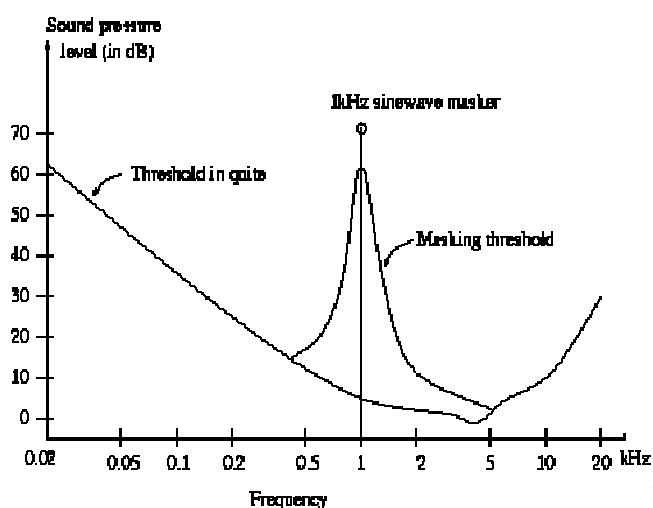


圖 1：1kHz 遮罩曲線

在 1kHz 單頻音的影響下，我們將人耳在各頻率所能夠聽到與不能聽到的音量界限用一條曲線描繪成圖 2 的狀況，這條曲線稱為遮罩曲線 (Masking Threshold)，在曲線下方的部分是人耳聽不到的，當我們在進行音訊的編碼時可以將被遮罩的這部份丟棄，以減少音訊編碼的資料量。

如果產生遮罩的單頻音減小，相對的遮罩曲線也會往下降，不過到了一個程度之後，不論遮罩音再如何減小時，遮罩曲線都不再變化，成為圖 2 虛線的情形，而這現象的產生解釋成為人耳聽覺的不均勻性。聲音的音量只要超過 3dB，人類的大腦神經便可以感覺到聲音的變化，而一般來說，人類的聽覺範圍通常以 20Hz—20KHz 為界，稱之為音頻，數位音訊在做壓縮處理的時候，可以先除去音頻以外的資料，同時受限於人耳敏感度的不均勻所無法察覺的部份也可以捨去。

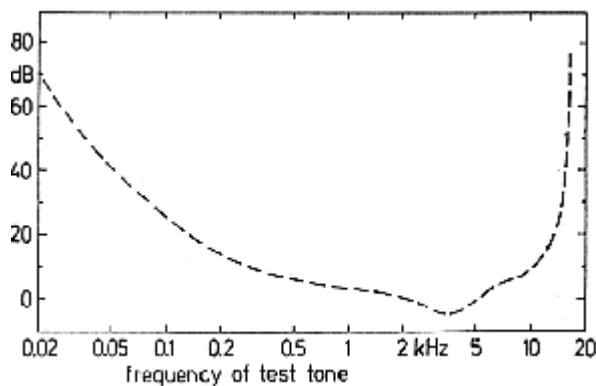


圖 2：人耳遮罩曲線

在現實的情況下，通常遮罩音不會單單只有一個，當很多頻率的遮罩音同時出現時，遮罩曲線就便得很複雜，就如同圖 3 的樣子。更難的是，遮罩曲線會隨著頻率、音量大小以及時間的變化有所不同，當這幾個條件有所改變時，遮罩曲線亦將不同，所以當音訊壓縮時，每取一段聲音資料就要更新遮罩曲線。

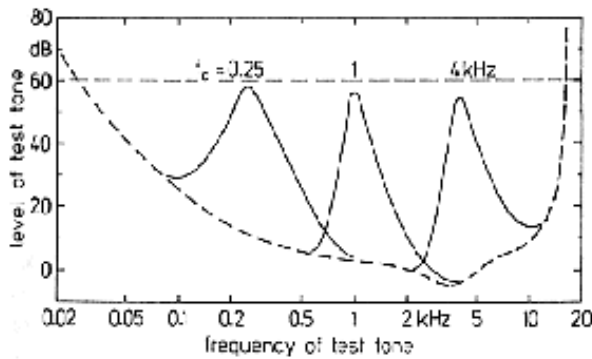


圖 3：數個遮罩曲線重疊情形

(2) 時間軸上的遮罩性：不僅遮罩效應會發生在頻率軸上，在時間軸上同樣的也會發生，當一段大音量聲音後，緊接來的是較小音量時，這後來的小音量則不易被察覺。一般解釋為神經細胞的適應造成的聽覺遲滯現象，甚至當小音量之後緊接著來大音量，也會有部份小音量的聲音會被後來的大音量給遮蓋住。不過目前利用時間軸的遮罩情形來減少資料量的音訊壓縮技術還不是很普遍。

Huffman code

Huffman code[2]是一種常見的無失真壓縮方案。當 PCM 訊號被分成好幾個頻段並經過以上的處理之後，最後經過 MDCT(Modified Discrete Cosine Transform)(類似 FFT(Fast Fourier Transforms))，將波型轉換為一連串的系數

Huffman code 的編碼過程須用到 2 元樹，同時也是 Huffman code 的解碼樹。一開始的做法是，把每個需要編碼的符號看做為一棵 2 元樹的樹葉，而每個樹葉有其加權值，加權值即出現的機率，而樹葉依加權值大小排列。

以下是編碼流程：

STEP 1 所有的樹葉為自由節點

STEP 2 從自由節點中找出加權值最小的兩個節點

STEP 3 為這兩個節點做一個父節點，其加權值為這兩個兒子節的加權值和

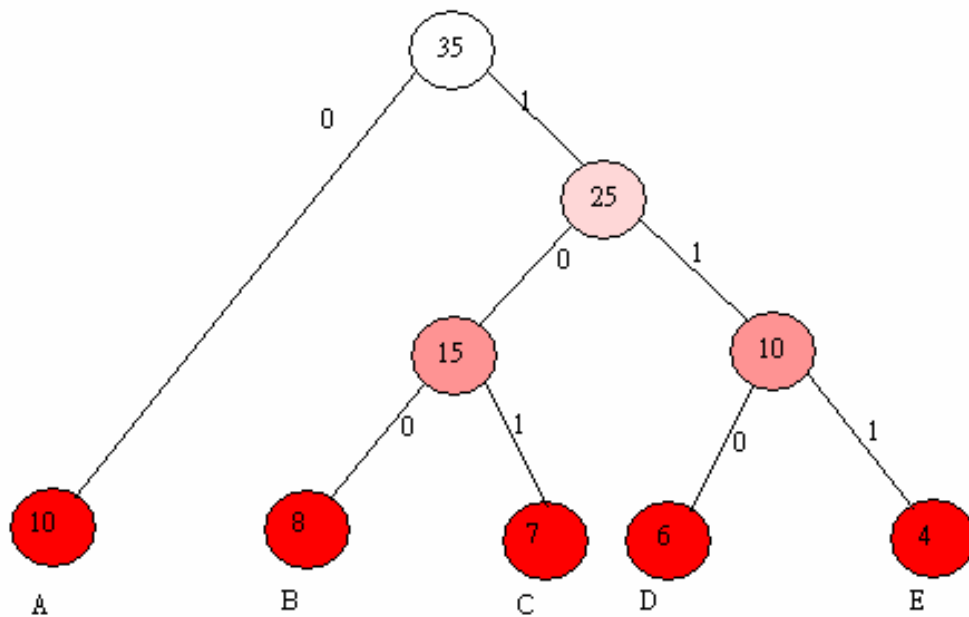
STEP 4 將父節點加入自由節點群，而兩個子節點從自由節點中除名

STEP 5 由父節點做樹枝到兩子節點，加權值大的給 1，小的給 0

STEP 6 重複 STEP2~STEP5，直到只剩一個自由節點(注意，不只同一階層)

舉例:有 A B C D E 5 個符號需要作編碼。

其出現的機率加權各為 A=10.B=8.C=7.D=6.E=4。



編碼的方式即從樹根走到樹葉即可。以上圖為例

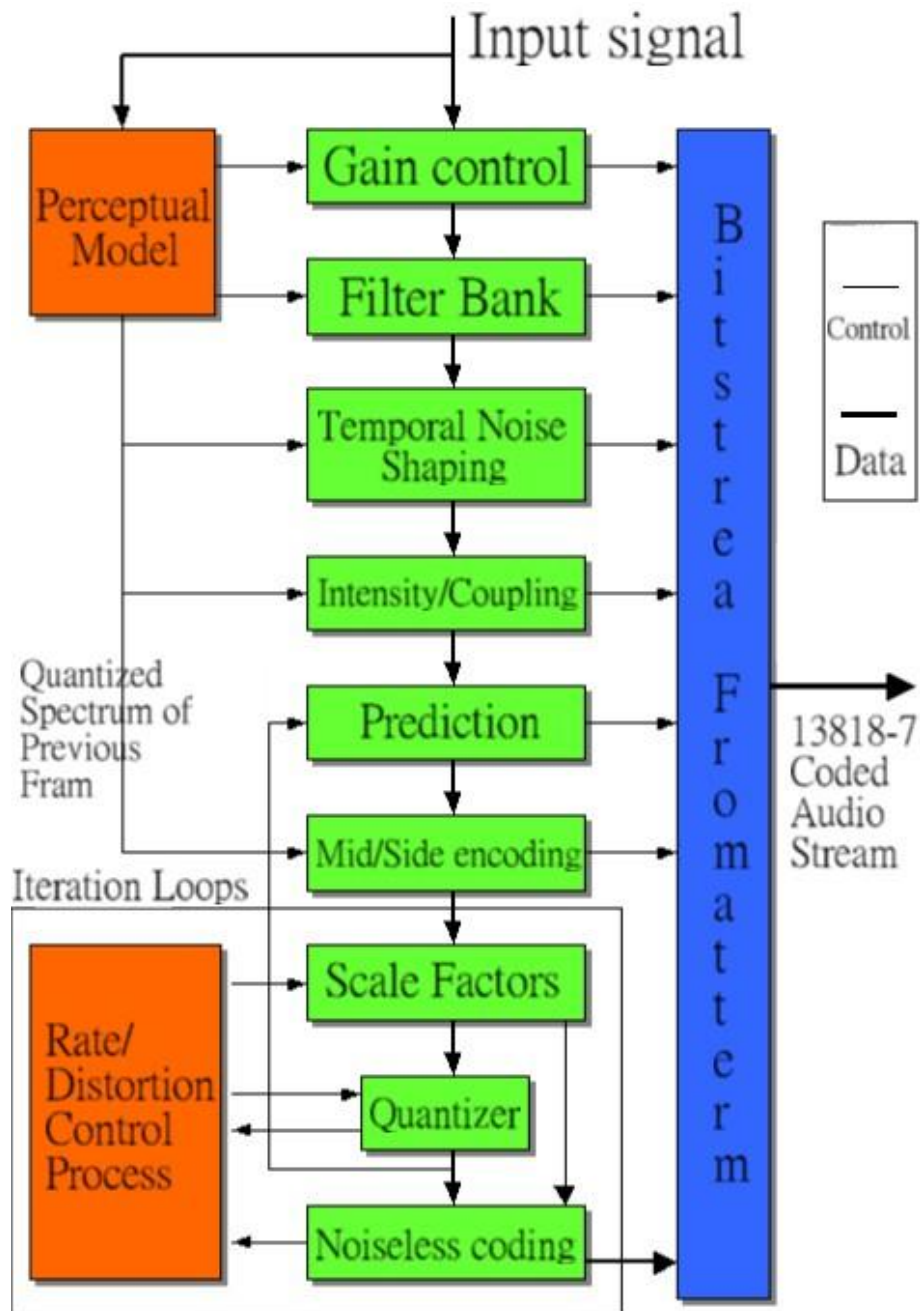
A=0.B=100.C=101.D=110.E=111。

三 AAC 歷史

MPEG 音訊壓縮規格在數位音訊的發展上佔有很重要的地位，而 MP3 正是應用此規格所發展出來的一種技術。MPEG 最早之標準稱為 MPEG-1，由國際標準化組織 (ISO) 在 1992 年完成，並制定成 ISO/IEC 11172 標準。從 MPEG-1 的規格上來看，設計的最大位元率 (bit rate) 達到 1.5Mbps，規格中分別規範了影像壓縮與聲音壓縮，VCD 就是使用 MPEG-1 的規格。MPEG-1 本身分成三個部份，包括了系統、影像與聲音資料。

隨著時間的推移，MP3 越來越不能滿足我們的需要了，比如壓縮率落後於 Ogg、WMA、VQF 等格式，音質也不夠理想，僅有兩個聲道……於是 Fraunhofer IIS 與 AT&T、Sony、杜比、諾基亞等公司展開合作，共同開發出了被譽為“21 世紀的資料壓縮方式”的 Advanced Audio Coding（簡稱 AAC）音頻格式，以取代 MP3 的位置。其實 AAC 的演算法在 1997 年就完成了，當時被稱為 MPEG-2 AAC，因為還是把它作為 MPEG-2 標準的延伸。但是隨著 MPEG-4 音頻標準在 2000 年成型，MPEG-2 AAC 也被作為它的編碼技術核心，同時追加了一些新的編碼特性，所以我們又叫 MPEG-4 AAC。但這兩者目前尚未被廣泛應用。

四 AAC 編碼流程圖[3]



1.增益控制(Gain control)

增益控制模組用在可變取樣率配置中，它由多相正交濾波器 PQF(polyphase quadrature filter)、增益檢測器(gain detector)和增益修正器(gain modifier)組成。這個模組把輸入信號分離到 4 個相等帶寬的頻帶中。在解碼器中也有增益控制模組，通過忽略 PQF 的高子帶信號獲得低取樣率輸出信號。[4]

2.濾波器組(Filter Bank)

AAC 使用的是 MDCT(Modified Discrete Cosine Transform) 濾波器組。AAC 的濾波器組被設計成允許視窗改變大小，用來適應輸入信號的狀態。視窗的大小隨著編碼器及解碼器同時改變，好讓濾波器組能有效率地分辨變化多端的輸入訊號。加上較長的轉換視窗長度，可變換的視窗型態，及可變更轉換區塊的長度，使得 MDCT 優於使用預先編碼法的濾波器組，並且提供濾波器組更好的頻率選擇性。雖然量化和編碼都是在頻域裡執行完，解碼濾波器組的功能是利用反 MDCT (IMDCT)，將解碼器輸入端頻譜值，轉換成時域的輸出值。對每個聲道而言，經由 IMDCT， $N/2$ 個的時間-頻率值 X_{ik} 被轉換到 N 個的時域值 x_{in} 裡。

MDCT 的表示法如下：

$$X_{ik} = 2 \sum_{n=0}^{N/2-1} x_{in} \cos\left[\frac{2\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right], k = 0, \dots, \frac{N}{2}-1.$$

IMDCT 的表示法如下：

$$x_{in} = \frac{2}{N} \sum_{k=0}^{N/2-1} X_{ik} \cos\left[\frac{2\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right], n = 0, \dots, N-1$$

此處 n 為樣本指標， N 為轉換視窗長度， i 為區指標，而 $n_0=(N/2+1)/2$ 。AAC 的濾波器組被設計成允許視窗改變大小，用來適應輸入信號的狀態。視窗的大小隨著編碼器及解碼器同時改變，好讓濾波器組能對變化多端的輸入訊號分離其頻譜成分。AAC 主要使用兩種視窗型態：正弦視窗 (sine window) 及凱撒貝索衍生視窗(Kaiser-Bessel Derived window)。AAC 允許濾波器組針對輸入信號的特性來改變時間頻率解析度。這個功能可藉著轉換輸入長度在 2048 或是 256 間的轉變樣本上完成。取長轉換 (2048 個樣本) 的優點是，在複雜的頻譜上信號編碼效率會提升，並且對短暫的靜態信號有較好的頻率解析度，然而，長轉換對急

遽變化的信號編碼卻沒什麼效率。但在轉變的區塊間轉換，卻在不同聲道間造成了時間調整上的問題。為了解決這個問題和確保在長區塊和短區塊轉換間有平滑的傳輸，在長短視窗切換間，使用了開始視窗 (start window) 及結束視窗 (stop window)。這個設計保存了 MDCT 和 IMDCT 的特性，並維持區塊的排列。

3. 時域雜訊修整(TNS, Temporal Noise Shaping)

在感知聲音編碼中，TNS 模組是用來控制量化雜訊的暫態形狀的一種方法，解決量化雜訊的錯誤匹配問題。這種技術的基本想法是，在時域中的音調聲信號在頻域中有一個暫態尖峰，TNS 使用這種雙重性來擴展已知的預測編碼技術，把量化雜訊置於實際的信號之下以避免錯誤匹配。

4. M/S 強度編碼(M/S Intensity Encoding)

在 SSR profile 裡，M/S 聯合聲道解碼機制將重建左右聲道的頻譜係數。M/S 立體聲編碼是用來控制並預測編碼時產生的雜訊。M/S 立體聲的開關狀態已經以信號位元的陣列被傳送至解碼器。當加強編碼被使用時，M/S 解碼機制將不會被啟動。在 AAC 解碼器中，LC profile 使用強度立體聲解碼和偶合聲道機制。強度立體聲解碼作用在右聲道傳進的信號並使用虛擬碼簿 INTENSITY_HCB(in-phase signal) 和 INTENSITY_HCB2(out-of-phase signal)。關於強度編碼資訊傳輸已表示在“強度立體聲位置”的值裡面，這個值指出左右聲道比例的關係。偶合聲道單元可用來實現強度型立體聲編碼並可將聲音物件動態地混合到一立體聲的訊號。所以附加註解的(commentary) 聲道可被加入一個已經存在的多聲道程式裡。

5. 量化(Quantizer)

AAC 的量化過程是使用兩個巢狀迴圈進行反覆運算。通過對量化分析的良好控制，bit rate 能夠被更高效地利用。

6.無雜訊編碼(Noiseless coding)

無雜訊編碼實際上就是霍夫曼編碼，它對被量化的譜系數、比例因數和方向資訊進行編碼。

7.數量因數(Scale Factors)

AAC 在頻譜係數群中使用個別放大率，稱為數量因數頻帶，來當作另一個方法以調整在頻域中的量化雜訊。數量因數表示一個增益值，用來改變頻譜係數在數量因數頻帶中的振幅大小。它的寬度是建立於模仿人體聽覺系統的關鍵頻帶 (critical band)。對所有的數量因數來說，前述的不同值皆是利用哈夫曼編碼。數量因數頻帶和與其相對應的係數是照著頻率高低排列。對每個非零或非強度立體聲編碼，頻譜的資料是經由哈夫曼編碼規格以四個或二個為一組復原回來。哈夫曼解碼規格會一直持續到這個段落裡所有的頻譜值都被解碼完。一旦所有的段落都已經解碼完成，這些資料就會和被解碼的數量因數相乘並恢復成編碼之前未交錯的樣子。

五 程式(program)

來源：<http://www.audiocoding.com/>

此程式使用 ISO/MPEG 2/4 AAC Encoder Library V1.24

Interface description:

ISO/MPEG 2/4 AAC Encoder函式庫提供了一個高階的介面給MPEG2以及MPEG4 ISO AAC檔案使用。

其主要的標頭檔由C/C++程式所定義出來:faac.h 函式原型

編碼器的核心存在於作業系統中的靜態連結函式庫，於Microsoft Windows中為libfaac.lib(UNIX中為libaac.a)

Usage:

Calling sequence-以下為編碼時的呼叫順序

編碼所需要的每一個程序都呼叫 `faacEncOpen()`

呼叫 `faacEncGetCurrentConfiguration()`來取得編碼器的參數，此函數採用C/C++的指標法回傳參數。然後再呼叫`faacEncSetConfiguration()`，來設定各個參數值。

只要有樣本需要被編碼的話， 呼叫`faacEncEncode()`將立即進行編碼工作，編碼完後會送出一連串位元流(bitstream)，放置於client-supplied buffer。

當你以零位元輸入來呼叫`faacEncEncode()`時，flushing程序將會被初始；之後你可以僅呼叫`faacEncEncode()`而零輸入，該函數會一直輸出編碼後之位元流，直到所有樣本都被編碼完。

一但 `faacEncEncode()`回傳結束位元時，立即呼叫`faacEncClose()`來結束整個編碼的程式。

Initialization / De-initialization

`faacEncOpen()`-

Prototype

`faacEncHandle FAACAPI faacEncOpen`

`(`

`unsigned long sampleRate,`

```
unsigned int numChannels,  
unsigned long *inputSamples,  
unsigned long *maxOutputBytes  
);
```

Description

開啓並初始編碼器的程序。

Parameters

sampleRate

代表輸入資料的取樣率。

numChannels

代表輸入資料的通道數目。

inputSamples

代表要供給各呼叫中**faacEncEncode()**的樣本。

maxOutputBytes

代表**faacEncEncode()**呼叫結束後，輸出buffer所收到的最大位元流。

Return value

若是發生錯誤產生，則回傳NULL。

faacEncClose()

Prototype

```
void FAACAPI faacEncClose  
(  
faacEncHandle hEncoder  
);
```

Description

關閉編碼器的程序。

Parameters

hEncoder

代表**faacEncOpen()**傳回的編碼器處理參數。

Encoder configuration

faacEncGetCurrentConfiguration()

Prototype

```
faacEncConfigurationPtr FAACAPI  
faacEncGetCurrentConfiguration  
(  
    faacEncHandle hEncoder  
);
```

Description

獲得此編碼器組態的結構性指標參數，可以改變設定這些參數並且傳給
faacEncSetConfiguration()使用。

faacEncSetConfiguration()

Prototype

```
int FAACAPI faacEncSetConfiguration  
(  
    faacDecHandle hDecoder,  
    faacEncConfigurationPtr config  
);
```

Description

視faacEncGetCurrentConfiguration()而定，來設定編碼器的組態。

Encoding functions

faacEncEncode()

Prototype

```
int FAACAPI faacEncEncode  
(  
    faacEncHandle hEncoder,  
    short *inputBuffer,  
    unsigned int samplesInput,  
    unsigned char *outputBuffer,  
    unsigned int bufferSize  
);
```

Description

對一個frame的樣本來進行編碼。

Parameters

hEncoder

代表編碼器處理參數。

inputBuffer

代表要編碼的樣本。

samplesInput

代表faacEncOpen()中所取得的樣本，將放於此處。只要該樣本是有效的，則以零輸入呼叫faacEncEncode()時，flushing程序將會起始。

outputBuffer

代表一個指標，指向編碼後bitstream存放的buffer，該buffer容量最小要超過faacEncOpen()中maxOutputBytes的大小。

Return value

當錯誤發生時，則有效位元將回傳負值。若是編碼成功則有效位元回傳0值。

Data structures reference

faacEncConfiguration

Definition

```
typedef struct faacEncConfiguration
{
    unsigned int mpegVersion;
    unsigned int aacObjectType;
    unsigned int allowMidside;
    unsigned int useLfe;
    unsigned int useTns;
    unsigned long bitRate;
    unsigned int bandwidth;
}
faacEncConfiguration, *faacEncConfigurationPtr;
```

Description

透過這結構化參數，可以改變編碼器的組態。

Fields

- mpegVersion

代表該編碼器MPEG版本，其中不是MPEG2就是MPEG4。

- **aacObjectType**

代表該AAC目的檔之型態。有以下三種: **MAIN**、**LOW**、**LTP**。

- **allowMidside**

代表mid/side coding方法使用與否。設為**1**時，則允許使用mid/side coding，反之設為**0**則不使用mid/side coding。

- **useLfe**

代表LFE channel的數量，如設為**1**則表示一個LFE channel。不過該功能還未提供使用。

- **useTns**

代表TNS 方法使用與否。設為**1**時，則允許使用TNS，反之設為**0**則不使用TNS。

- **bitRate**

代表一個channel的位元率。

- **bandwidth**

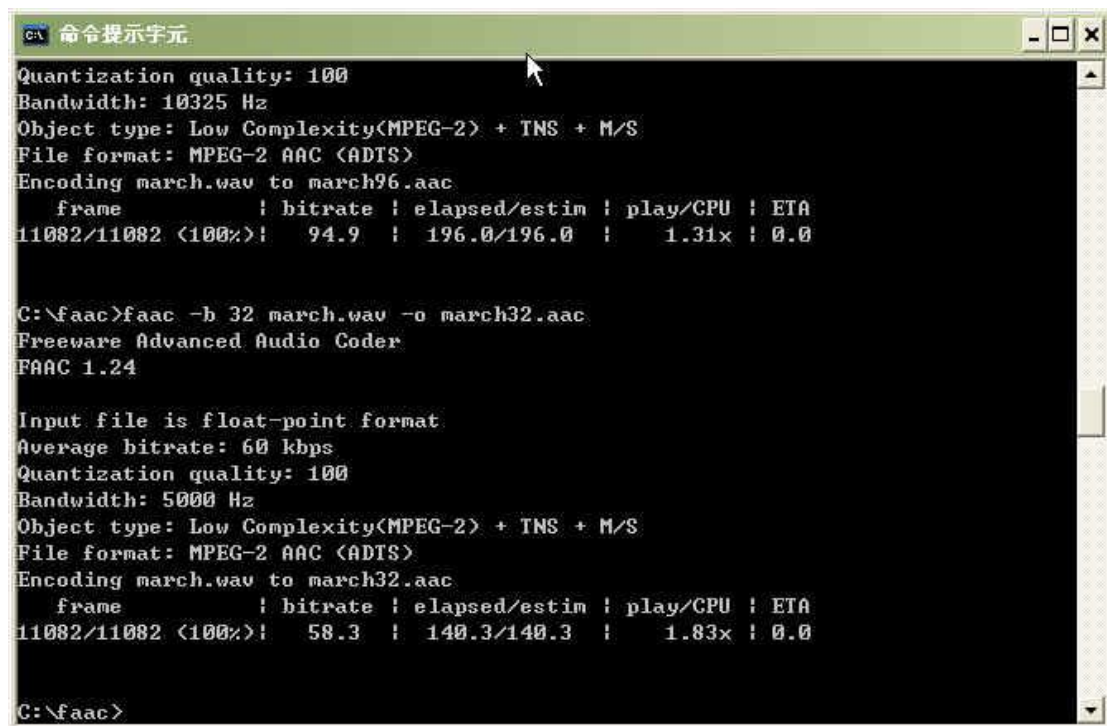
代表最大的頻寬。單位Hz。

六 AAC 格式實作

我們做了 AAC 格式與 MP3、WMA、OGG 這三種格式的比較壓縮率

編碼用軟體：1.FAAC1.24

來源：<http://www.audiocoding.com/>



```
命令提示字元
Quantization quality: 100
Bandwidth: 10325 Hz
Object type: Low Complexity(MPEG-2) + TNS + M/S
File format: MPEG-2 AAC (ADTS)
Encoding march.wav to march96.aac
   frame      | bitrate | elapsed/estim | play/CPU | ETA
11082/11082 <100%>|  94.9   | 196.0/196.0   |  1.31x   | 0.0

C:\faac>faac -b 32 march.wav -o march32.aac
Freeware Advanced Audio Coder
FAAC 1.24

Input file is float-point format
Average bitrate: 60 kbps
Quantization quality: 100
Bandwidth: 5000 Hz
Object type: Low Complexity(MPEG-2) + TNS + M/S
File format: MPEG-2 AAC (ADTS)
Encoding march.wav to march32.aac
   frame      | bitrate | elapsed/estim | play/CPU | ETA
11082/11082 <100%>|  58.3   | 140.3/140.3   |  1.83x   | 0.0

C:\faac>
```

執行方式：faac -b<輸入 bit rate> 來源聲音檔.wav -o 輸出檔名.aac

例：faac -b 128 march.wav -o march128.aac

2.Advanced MP3 Converter2.18



來源：<http://www.mp3do.com/>

功能:可將 WAV 檔轉成 MP3、WMA、OGG 之格式

3.Winamp5.0

來源：<http://www.winamp.com/>

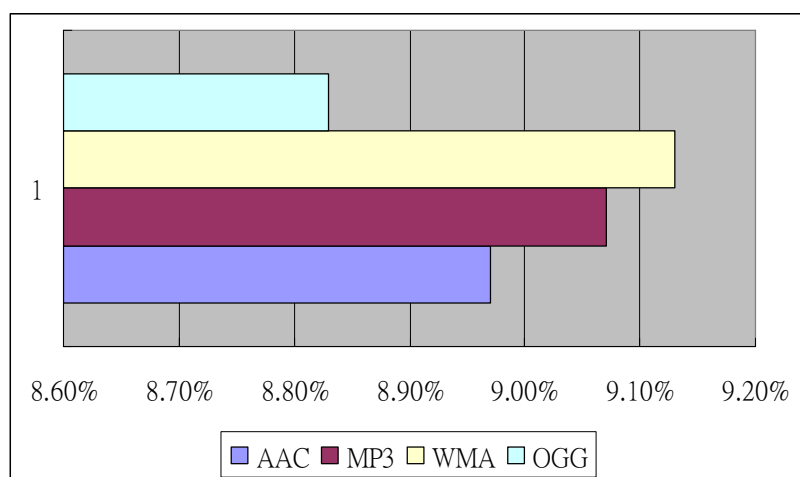
功能：播放 AAC、MP3、WMA、OGG 之格式音樂

我們分別對不同的 bit rate 做壓縮率比較 (Sampling rate 皆為 44100Hz)

WAV 大小：44323KB

128Kbps (檔名 march128)

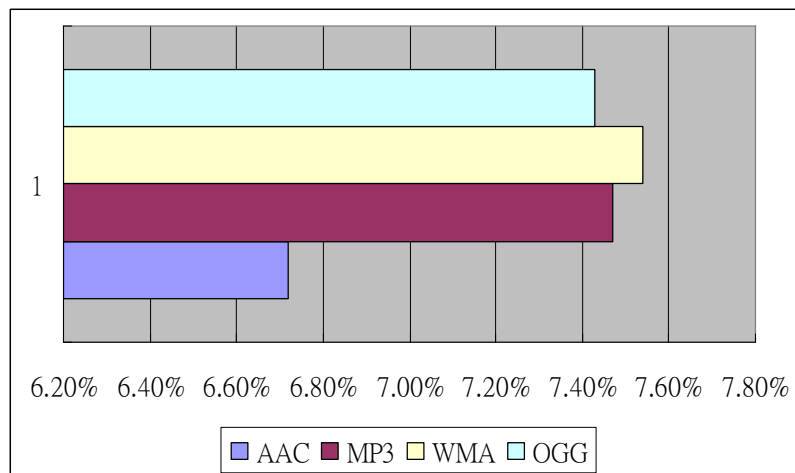
格式	壓縮後大小	壓縮率
AAC	3975KB	8.97%
MP3	4018KB	9.07%
WMA	4047KB	9.13%
OGG	3912KB	8.83%



96Kbps : (檔名 march96)

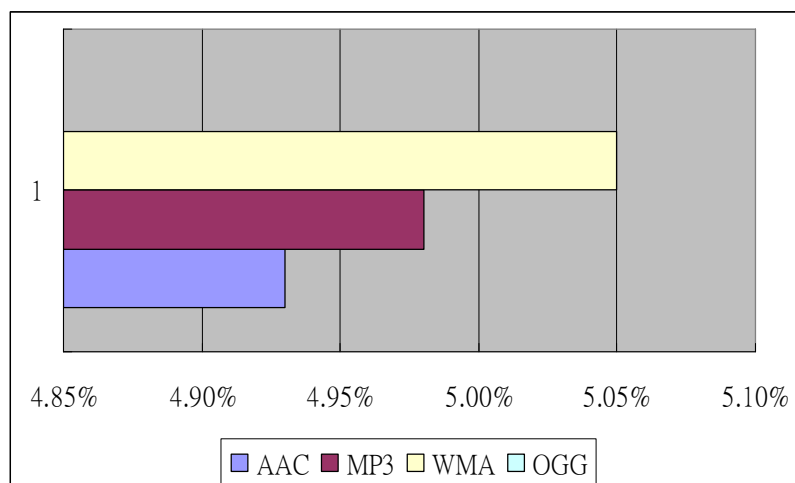
格式	壓縮後大小	壓縮率
AAC	2982KB	6.72%

MP3	3014KB	7.47%
WMA	3042KB	7.54%
OGG	2996KB	7.43%



64Kbps : (檔名 march64)

格式	壓縮後大小	壓縮率
AAC	1989KB	4.93%
MP3	2009KB	4.98%
WMA	2036KB	5.05%
OGG		



實作二

分別對 aac 及 mp3 格式做比較

曲一 The Four Season - Allegro

WAV 大小 : 35338KB

Bitrate 格式	128	96	64
MP3	3209KB	2407KB	1604KB
AAC	3210KB	2407KB	1604KB

曲二 望春風

WAV 大小 : 43577KB

Bitrate 格式	128	96	64
MP3	3953KB	2965KB	1977KB
AAC	3926KB	2945KB	1965KB

曲三 蔡依林 - 愛情三十六計

WAV 大小 : 37126KB

Bitrate 格式	128	96	64
MP3	3367KB	2525KB	1683KB
AAC	3367KB	2524KB	1682KB

七 未來展望

目前支援 AAC 的產品還比較少，這主要是因為專利使用費大大限制了 AAC 的發展！不過好在有諾基亞、蘋果、松下三大巨頭的鼎力支持，場面還不算冷清。



N—Gage



松下 SV—SD50

圖檔來源：<http://www.yesky.com>

未來專利使用費降低後，相信 AAC 格式一定會大受歡迎。

八 結論

AAC 其實是一種高壓縮比的音頻壓縮演算法，它的壓縮比遠遠超過了較老的音頻壓縮演算法，如 MP3。AAC 和 AC-3 都是變換編碼演算法，但 AAC 使用了解析度更高的濾波器組，因此它可以達到更高的壓縮比。另外 AAC 還使用了時域雜訊修整和量化哈夫曼編碼等最新技術，這些新技術的使用都使壓縮比得到進一步的提高。而且，AAC 比 AC-3 更靈活，它支持更多種取樣率和 bit rate、支持 1 個到 48 個音軌、支持多達 15 個低頻音軌、具有多種語言的相容能力、還有多達 15 個內嵌資料流程。

這次的實作讓我們了解到的新一代音樂格式的威力，或許不久的將來，路上人手拿的會是 AAC 隨身聽，而不再是現下最流行的 MP3 了。

九 參考文獻

- [1] "Mpeg Digital Audio Coding" , IEEE Signal Processing Magazine, 1997-9
- [2] Joan L. Mitchell, William B. Pennebaker, Chad E. Fogg, and Cidier J.LeGall, "MPEG VIDEO COMPRESSION STANDARD" , 1996
- [3] Marina Bosi, Karlheinz Brandenburg, Schuyler Quackenbush, Louis Fielder, enzo Akagiri, Hendrik Fuchs, Martin Dietz, Jorgen Herre, Grant Davidson, Yoshiaki Oikawa, "ISO/ICE MPEG-2 Advanced Audio Coding" , 1997
- [4] Watkinson, John, "The MPEG handbook :/MPEG-1, MPEG-2, MPEG-4" ,2001
- [5] Bosi Metal. ISO/IEC MPEG-2 Advanved Audio Coding. Journal of the AudioEngineering Society, No. 10, Oct 1997, pp789-813