行政院國家科學委員會專題研究計畫　成果報告

子計畫一：無線網路串流聲訊研究及聲視訊子系統整合

(2/2)

計畫類別：整合型計畫
計畫編號：NSC93-2219-E-009-006-
執行期間：93 年 08 月 01 日至 94 年 07 月 31 日
執行單位：國立交通大學電子工程學系暨電子研究所

計畫主持人：杭學鳴

計畫參與人員：楊政翰，陳繼大，王盈閔，陳志楹，林鴻志，黃育彰，黃祥哲

報告類型：完整報告
報告附件：出席國際會議研究心得報告及發表論文
處理方式：本計畫可公開查詢

中　華　民　國 94 年 10 月 31 日

行政院國家科學委員會補助專題研究計畫 ■ 成 果 報 告
□期中進度報告

基於正交分頻多重進接之無線多媒體傳收機研究及設計(2) -- 子計畫一：無線網路串流聲訊研究及聲視訊子系統整合 (2/2)

Wireless Streaming Audio Research and Audio/Video Subsystem Integration (2/2)

計畫類別：□ 個別型計畫　　■ 整合型計畫

計畫編號： NSC 93-2219-E009-006

執行期間： 93 年 8 月 1 日至 94 年 7 月 31 日

計畫主持人：杭學鳴

計畫參與人員：楊政翰，陳繼大，王盈閔，陳志楹，林鴻志，黃育彰，黃祥哲　交通大學電子工程學系 研究生/博士後

成果報告類型(依經費核定清單規定繳交)：□精簡報告　■完整報告

本成果報告包括以下應繳交之附件：

□赴國外出差或研習心得報告一份

□赴大陸地區出差或研習心得報告一份

■出席國際學術會議心得報告及發表之論文各一份

□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢

□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：國立交通大學電子工程學系

中　華　民　國　94 年　10 月　31 日

行政院國家科學委員會專題研究計畫成果報告

基於正交分頻多重進接之無線多媒體傳收機研究及設計(2) -- 子計畫一：無線網路串流聲訊研究及聲視訊子系統整合（2/2）

Wireless Streaming Audio Research and Audio/Video Subsystem Integration (2/2)

計畫編號: NSC 93-2219-E009-006
執行期限: 93 年 8 月 1 日至 94 年 7 月 31 日

主持人: 杭學鳴　國立交通大學電子工程學系教授
計畫參與人員：楊政翰，陳繼大，王盈閔，陳志楹，林鴻志，黃育彰，黃祥哲
交通大學電子工程學系 研究生/博士後

國立交通大學電子研究所

中文摘要

本子計畫之主要目標在研究與製作寬頻無線網路環境中的串流聲訊系統,包含聲訊和音訊壓縮之編碼與解碼，以及錯誤控制編碼。本年度完成此部分在 DSP 上的實現，並將之加速以符合無線網路頻寬上的要求。在音訊壓縮部分，本計畫採用 MPEG-4 AAC。在編碼端的計算中，心理聲學模式計算以及位元編碼(Bit allocation)所佔的執行時間為最大。我們將心理聲學模式的計算所需的 FT 改為 MDCT，進而減少了 FFT 的計算；此外，亦更改位元編碼的計算方式，將原本兩層的迴圈簡化為單層。經由加速後，最後的編碼器版本在 DSP 上執行速度的改善幅度達 78%，並已達到即時演算的要求。在語音壓縮部分，本計畫採用 AMR(Adaptive Multi-Rate)。在利用 DSP 本身的內建函式加速後，編碼器最終的處理速度為 14.05ms/frame。在錯誤控制編碼方面，Reed-Solomon 編碼(Reed-Solomon Code)中 Syndrome 計算(Syndrome computation)與 Chien 搜尋法(Chien Search)的部分所花費時間最多。為此，我們使用 FFT radix-4 的方法來對原本的 Syndrome 計算公式做整理，使其在一個迴圈中可以處理四組獨立運算。另外 Chien 搜尋法方面，我們利用 BRS(Berlekamp-Rumsey-Solomon)解根演算法來降低伽羅瓦場(Galois Field)乘法的使用量。再加上 DSP 的技巧後，解碼器最後可以達到 176.4Kbytes/sec 的處理速度。

關鍵詞：MPEG，DSP，音訊壓縮，語音壓縮，AAC，ARM，心理聲學模式，位元編碼，Reed-Solomon 編碼，Syndrome 計算，Chien 搜尋法

Abstract

The goal of this research project is to study, simulate and design effective streaming speech and audio algorithms/systems transmitted in the wideband wireless environment on DSP platforms. The topics under study are speech and audio compression, and 802.16a error control codes. The adopted codec for audio compression is MPEG-4 AAC and the codec for speech compression is AMR (Adaptive Multi-Rate). This year, three tasks have been implemented on DSP, and we accelerate them to meet the requirements of wideband wireless transmission. In the audio compression part, the two heaviest computational units in the AAC encoder are the psychoacoustic model and the bit allocation algorithm. To speed up the encoder, the psychoacoustic model computation is modified and thus MDCT coefficients are used to replace the Fourier transform coefficients without sacrificing accuracy. The bit allocation algorithm is simplified from a two-loop structure to a one-loop structure. The improvement in the final version is 78% and thus the real-time computation requirement is achieved. In the speech compression codec part, the C code of AMR is modified by using the TI intrinsic functions, and, at the end, the computation speed is improved to 14.05ms/frame. In the error control coding part, the Syndrome Computation and the Chien Search are the most time-consuming components in a Reed-Solomon Coder. The Syndrome Computation unit is modified by using the FFT radix-4 technique to enable four independent parallel computation processes in one loop. For computing the Chien Search, the BRS Algorithm is applied to reduce the Galois Field multiplications. The final decoder with the above and other modifications can reach the rate of 176.4Kbytes/sec.

關鍵詞：MPEG, DSP, AMR, Audio Compression, AAC, Psychoacoustic model, Bit allocation, Reed-Solomon Code, Syndrome computation, Chien Search

目錄 Table of Contents

A. Part 1: Audio Coding

A.1 文獻探討

AAC 是 MPEG-2 中的音訊編碼標準，於 1997 制定完成。MPEG-2 AAC 聲訊編碼標準捨棄與 MPEG-1 聲訊編碼標準的相容性,加入了時域雜訊重整 Temporal Noise Shaping (TNS) 及預測 Prediction 這兩個獨立的新模組，因此 AAC 能提供比 MP3 更好的壓縮率及聲訊品質 [1] [2]。MPEG-4 AAC Version 2 是 ISO/IEC MPEG 於 1999 年制定完成之新一代聲訊編碼標準，架構如圖 A.1所示。MPEG-4 AAC 聲訊編碼是以 MPEG-2 AAC 為基礎，並加入了數個獨立的新模組，長期預測 Long Term Prediction (LTP)、感知雜訊替代 Perceptual Noise Substitution (PNS)、Transform-Domain Weighted Interleave Vector Quantization (Twin-VQ)等。這些新的模組將有助於更低位元率的聲訊壓縮。

本計畫使用德州儀器 C6416 這顆 DSP 作為實現平台，同時把 AAC 編解碼器實現在 II(Innovative Integration) 所提供的 Quixote DSP 平台上,並且針對需大量運算的部分做加速。

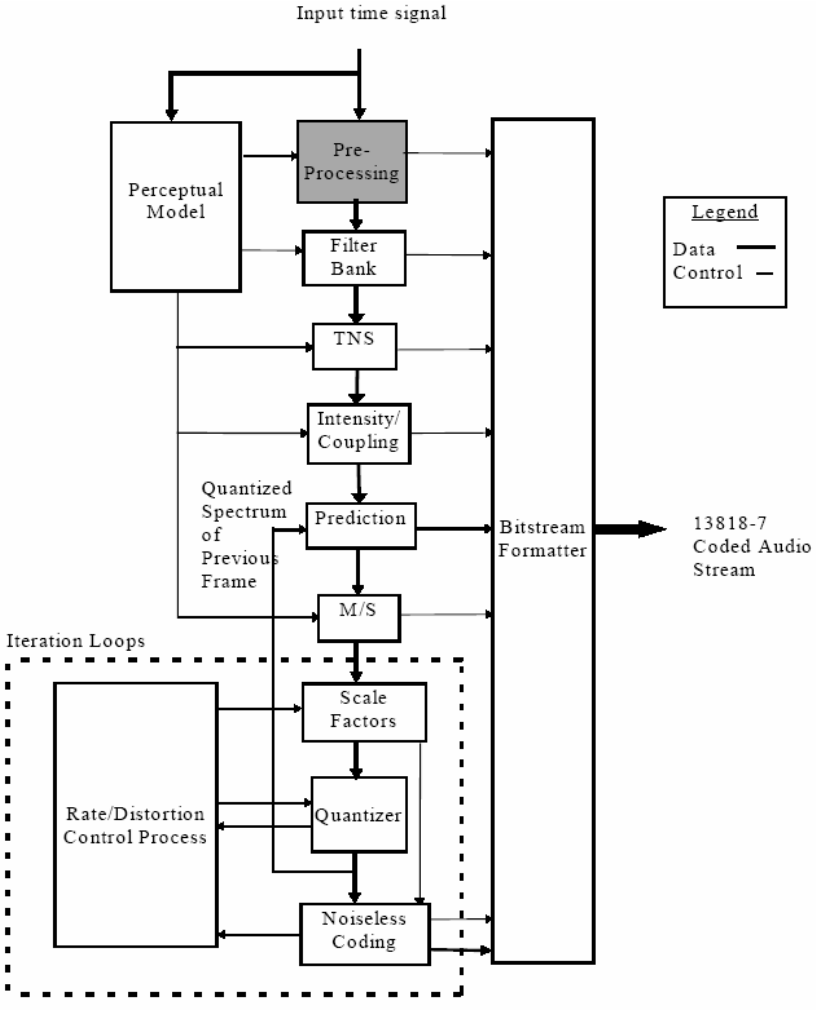


圖 A.1 MPEG-4 AAC 聲訊編碼整體架構

A.2 研究方法

我們為了將程式放在DSP上，因此統計AAC編碼器在DSP上執行的時間，結果如圖 A.2所示。

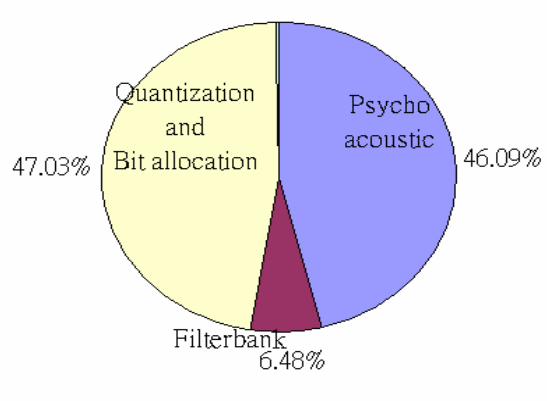


圖 A.2 AAC 執行效率方塊圖

根據圖 A.2的結果，我們可以發現心理聲學模式(Psycho-acoustic mode)以及位元編碼(Bit allocation)所佔的執行時間最多，因此我們參考了一些快速演算法來做改善。

A.2.1 心理聲學模式 Psycho-acoustic model

心理聲學模式會對輸入的音訊資料做分析並且決定出下一級的量化編碼時所產生的量化雜訊可以允許到哪種程度，而利用此資訊可以在有限制的編碼位元數來做編碼，圖 A.3為心理聲學模式之流程圖。

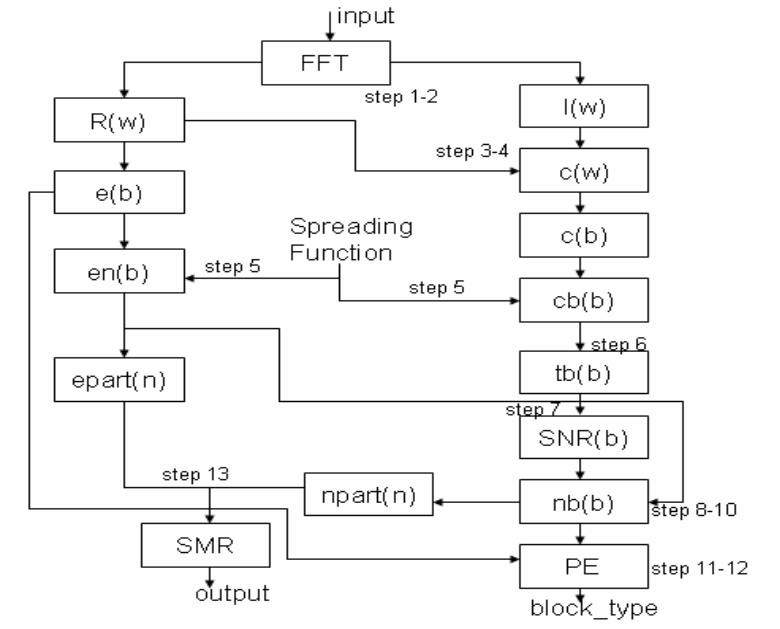


圖 A.3心理聲學模式之方塊圖

根據[3]，我們利用比較快速的演算法來改善心理聲學模式，此演算法利用已經在 Filterbank 處理方塊的修正離散餘弦轉換 MDCT 取代快速頻域轉換 FFT，並且也利用建表法來處理展開函式 Spreading function 的運算，如圖 A.4 所示。

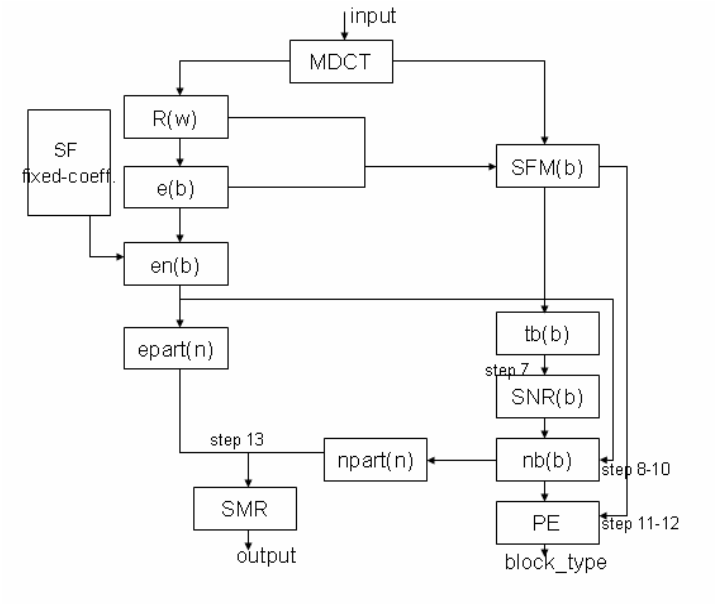


圖 A.4 快速演算法之心理聲學模式

A.2.2 位元編碼 Bit Allocation

位元編碼主要是在有限制的位元數並且編碼中所造成的量化雜訊合乎允許下，對資料做最佳效果編碼的工作。它主要有兩個迴圈，外迴圈主要是控制量化雜訊，而此量化雜訊是從內迴圈的頻域資料的量化所造成，而內迴圈主要是做量化及編碼的工作，如圖 A.5 所示。因此要在有限的位元數編碼允許下且減少量化雜訊目標下，常常因為重複執行兩個迴圈而導致龐大計算量。

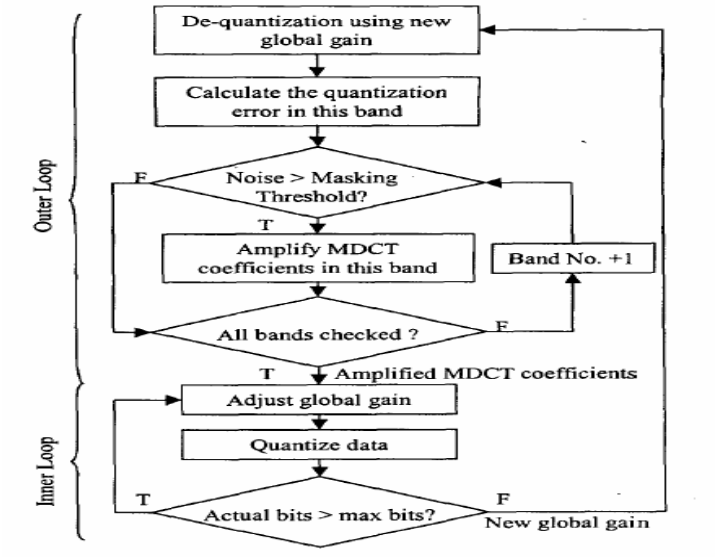


圖 A.5 位元編碼之方塊圖

根據[4]，我們加速了外迴圈，使它只需要執行一次迴圈即可。而此快速演算法是利用估計雜訊的原理，去計算出我們所要的 scale factor 值，進而使內迴圈在編碼時所產生的量化雜訊合乎標準。此快速演算步驟如圖 A.6 所示。

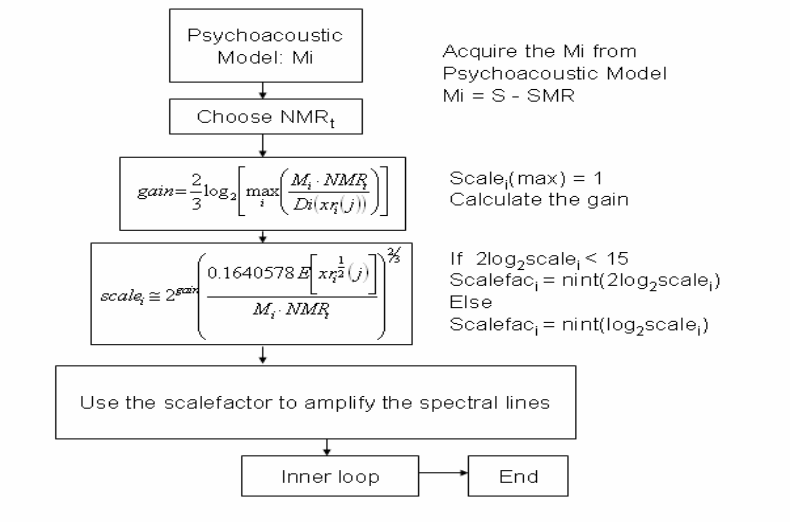


圖 A.6 快速演算法之位元編碼

A.3 實驗與結果

A.3.1 在 DSP 上的模擬結果

我們在 DSP 上的模擬下，表 A.1是效能的改善，其中(b)是針對程式架構以及在 DSP 上的環境設定的修改所改善的幅度，(c)則是加上演算法的改良後，最後所得出的效能改善。

	Total Execution Cycles	Performance Improvement
(a) Original	2,126,810,017	
(b) Code Acceleration	1,627,141,833	23.5 %
(c) Final	470,273,769	77.89 %

表 A.1 在 DSP 上的程式加速結果

表 A.2則是主要函式跟原來相比後的效率改善幅度。

Function	Improvement (%)
Total	77.89
Psycho-acoustics	89.09
Filterbank	34.55
Quantization and Bit-allocation	73.52

表 A.2 主要函式在 DSP 上的加速結果

A.3.2 DSP 平台上的實現結果

最後我們把 AAC 編碼器實現在 DSP 板子上去執行，所得的結果如表 A.3，其中(b)是開啟 CCS 的一些最佳化設定，(c)是加上程式架構修改後的結果，(d)則是再加上演算法改良後最後的結果。經過加速後，此程式在 DSP 平台上的速度可以達到即時 real-time 的效果。

	(a)	(b)	(c)	(d)
AAC en-coder	Without open opt. level	Open opt. level (file level)	Code Accelera-tion	Final implementation result
Time (s/frame)	0.1742	0.13925	0.08724	0.008

表 A.3 實現結果

A.4 結論

我們為了在 DSP 平台上加速先進音訊編碼器，因此針對 DSP 的架構使用了一些程式技巧，包括 fixed-point 資料型態、TI DSP 的特殊指令等等。除此之外，我們也參考了一些快速運行的演算法，並套用在原來的音訊編碼器之心理聲學模式及量化位元編碼上。經由這些的程式修改，最後的編碼器版本在 DSP 上的執行速度比原來的有了 77.89%的改善幅度。並且我們也成功的把先進音訊編碼器及解碼器兩者實現在 II(Innovative Integration) 所提供的 Quixote DSP 平台上。而在主端及客端的傳輸介面，我們採用了緩衝之區塊傳輸模式，此模式讓我們容易實現整個架構。最後經由我們的加速及系統實現，此先進音訊編碼及解碼器各自都可達到即時編解碼的效果。

B. Part 2: Speech Coding

B.1 研究目的

無線通訊在目前電子產業中已漸趨普及，其中於多媒體方面的應用更是愈來愈重要，現今多媒體的應用主要包括視訊、音訊和語音，其品質主要取決於本身訊源壓縮的效率，希望能在有限的傳輸速率或是容易受到雜訊干擾的環境下，盡可能降低失真的程度，便是此部分主要的研究方向之一；在語音部分，3GPP 制訂了 Adaptive Multi-Rate(AMR)語音編碼標準[5][6]，規定了八種不同的傳輸速率下去做編碼。目的在於能夠根據各種不同的傳輸環境，來調整語音與錯誤修正編碼之間的比例，以有效率的提升語音品質，同時降低傳輸環境所造成的影響。G.723.1 為 3GPP 所制訂的前一代語音標準，在相似的傳輸速率下，藉由 AMR 編碼可以得到較 G.723.1 佳的語音品質。

本研究部分所使用的 AMR reference software 為 3GPP 所提供，使用 C++語言撰寫，我們最終目的要將 AMR 的編碼與解碼器實現至德州儀器 C6416 的 DSP 平台上。因此我們的主要工作首先要考慮此實現平台的硬體架構，藉此減低程式的複雜度。待其運算速度達到 real time 的目標之後，進而實現至 DSP 平台上，使得程式能夠更有效率的執行，同時也盡可能發揮硬體的最大功效。

B.2 文獻探討

參考 TI 出版之 Programmer's Guide[7]，TI 所建議的 DSP 程式設計及最佳化流程可先進行程式的 Profile，也就是利用 TI DSP 專用之程式編譯器 Code Composer Studio (CCS)進行編譯後。執行內附之 Profiler 功能後，分析所產生之檔案 Profile，觀察出程式中各個部分功能運行時所消耗的資源(或是運算時間)，進而對佔據最多運算時間的部分來做改善。

一般來說，初步從演算法著手可以得到較為顯著的改善，接著可以配合 DSP 的硬體架構，繼續對 C 語言的撰寫方式去做改進，使得編譯器能夠編譯出更有效率的程式。此方面可以利用 CCS 編譯器回報(Compiler's Feedback)部分，觀察到我們所設計的程式在 DSP 硬體中的資源使用情形，以及 TI DSP 特別支援的 Software Pipelining(SP)功能的運作情況，由資源使用的分佈平均與否以及 SP 的平行化效率，來判斷我們的程式撰寫是否可以適應 DSP 特殊的硬體架構，亦可使執行效率提升。

若是單單只有在 C 語言上去做調整，所能得到的改進幅度有限。在 TI 的 Programmer's Guide 中還有提供了線性組合語言(Linear Assembly)的層級來供程式撰寫者來做進一步的改善，雖然從組合語言著手能夠對每個指令做最佳的調整，以符合 DSP 的運算架構。但其層級相對於 C 較低，因此採用這種最佳化的手段是比較繁瑣而且耗費時間的，只有在以上方法皆無法使效能顯著提升時，才考慮對此部分做調整。

由於我們所使用的 reference software 是由 3GPP 所提供，其 C 語言的程式撰寫方式已做過某個程度上的最佳化，因此繼續對 C 語言做調整並非最有效率的改善方式。在 Programmer's Guide 中還提供了 intrinsic function。此為 TI C6000 系列編譯器所支援的特別指令，在編譯時會自動被取代為 C64 的組合語言指令，使得在不增加指令數目的情況下，善用 DSP 的特殊架構來做運算。此改善方法顯然能夠極有效率地降低程式在 DSP 上的執行時間，因此接下來便以此方式配合 C 語言細部的一些調整來做改善。

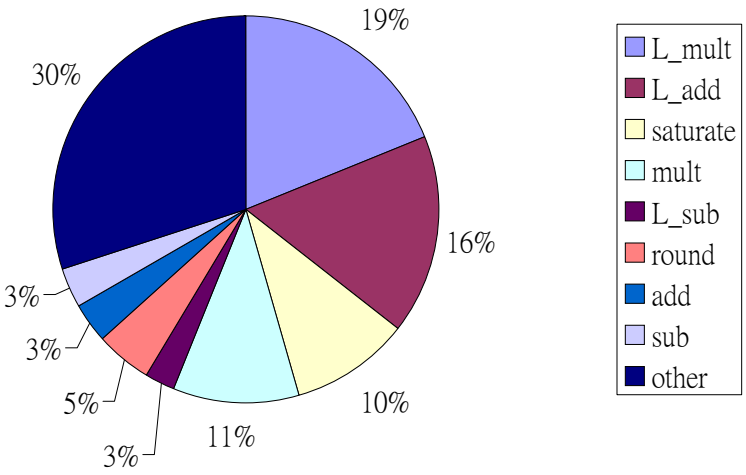


圖 B.1. 3GPP Reference Software 的 Profile 資料
(右方所列為前九個最耗運算時間的算數運算函數)

B.3 研究方法

B.3.1 改善程式運算效率

依據 Programmer's Guide 中所建議的改善步驟，我們先對 reference software 做 Profile，藉此得知哪一個部分需要最多的執行時間，進而從該部分著手改善。經過 Profile 後所得到的數據如圖 B.1，其中我們只將最耗運算時間的前九個函數列出，已佔了將近 70%的總執行時間，進一步觀察會發現這些函數皆為算數運算函數，也就是專門處理加法、減法、乘法或除法...等等的函數。觀察程式的細節也會發現，幾乎所有的函數都是由算數運算函數所組成，因此我們將從這些算數運算函數著手。但是由於這些算數運算函數所包含的指令極為簡單，我們很難在演算法與 C 語言的撰寫方式上做進一步調整。基於此原因以及綜觀「文獻探討」中討論各種改善方式的結果，使用 intrinsic function 將是最快速，且改善效果最顯著的方式。

雖然 TI DSP 提供了許多 intrinsic function，但 AMR 編碼程式中所使用的算數運算函數種類繁多。因此有時仍須對 intrinsic function 做些調整來達到我們所要的運算，舉例如下：

首先以 intrinsic function「_abs2」為例，此用於將兩個 16 位元的數值取絕對值並一同存入一個 32 位元的變數中。若我們希望將一個 32 位元的數值取絕對值並截取為一 16 位元的數值，我們便可以同時將 0 與此 32 位元的數值作為其輸入，再取其輸出數值的最低 16 位元即可達成我們所要的運算。

接下來假設要對某 16 位元的變數取負號，溢位(overflow)則做 saturation 的動作。我們知道，對某數取負號相當於用 0 減去該數值，因此我們可以使用 intrinsic function 中的「_ssub」—兩數相減並判斷是否溢位，溢位則做 saturation，將 0 減去此輸入的 16 位元數值。但是_ssub 的減法與判斷溢位皆為 32 位元運算，所以我們可以改為先將該 16 位元的輸入值向左位移成 32 位元的數值。待 0 與此 32 位元數值相減後，再將結果向右位移 16 位元，取其最低 16 位元即為我們所要的數值，經過調整之後的效果便會相當於我們要的運算。

另外，以函數「saturate」為例，即將 32 位元的輸入值截取為 16 位元，並判斷其是否溢位，若溢位則做 saturation。在此我們有兩種實現方式，第一，可以使用類似前例的實現方法，我們使用 intrinsic function 中的「_sshl」。其作用為將輸入數值向左位移所要的位元數，

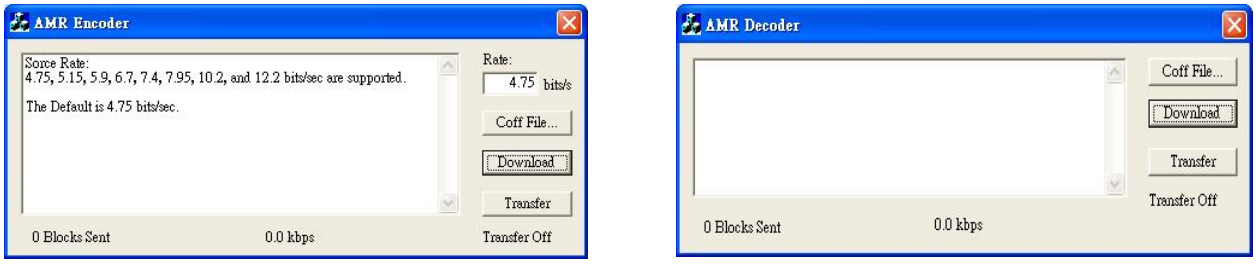
並根據溢位情形來決定是否做 saturation，亦為 32 位元運算。同理，我們可以先利用_sshl 將 32 位元的輸入值向左位移 16 位元，待其判斷是否做 saturation 後，再將輸出向右位移 16 位元，最低的 16 位元即為我們所要的結果。另一方法可使用「_spack2」來實現。此 intrinsic function 是用來將兩個輸入值截取為 16 位元，並分別判斷是否需做 saturation，最後再將個別的結果一起存入一個 32 位元的變數中。因此我們可以直接將 0 以及要做運算的 32 位元數值作為_spack2 的輸入，取_spack2 輸出的最低 16 位元即相當於我們所要的運算。

AMR 編碼與解碼器皆經過上述的方式加速，同時對 C 語言的撰寫方式做些微調後，我們進一步設定 CCS 的編譯器最佳化層級來對程式做改善，以使其更能配合 C6416 的 DSP 架構去做執行。

B.3.2 DSP 平台實現

最後，我們將 AMR 編碼器與解碼器實現至 C6416 平台上。此平台與電腦主機之間的傳輸介面可分為三種：Packetized Message Interface、Busmaster Interface 與 Block Mode Streaming。Packetized Message Interface 主要用於傳輸命令與訊號，由於其傳輸速率較低，一次可傳輸的資料量也較少，不適於用來傳輸運算資料。Busmaster Interface 為最快的傳輸模式，其適用於連續性的資料傳輸，也因此可以減少主機與 DSP 平台之間的訊號傳遞。但由於我們的 AMR 編碼與解碼器皆以一個框架(frame)為單位，所以在此我們選擇 Block Mode Streaming 的傳輸模式來做實現。其可支援不同大小的資料區塊(data block)為單位去做傳輸，由於可以一次傳送大量資料，因此亦為一有效率的高速傳輸模式。

程式架構方面，在此使用 GUI 介面來提供使用者對程式做操作，編碼器的操作介面如圖 B.2 (a)所示，首先需輸入編碼的傳輸速率，接著選定要下載至 DSP 平台上的程式並執行下載的動作，最後按下「Transfer」即開始執行程式。此外可在執行途中再次按下「Transfer」鈕以暫停運算，此時允許使用者更新編碼的傳輸速率，在下次執行時，接下來的語音框架便會依此新的模式來做編碼。解碼器部分的操作介面如圖 B.2 (b)所示，除了沒有編碼模式的輸入欄，可直接由編碼後的串流檔得知，其他皆與編碼器的介面相同。AMR 編碼器的程式執行流程大致為：主機端與 DSP 端完成初始化與下載程式的動作之後，便進入主要的工作部分，主機端從要做編碼的語音檔中擷取一個框架，再利用 Block Mode Streaming 的傳輸方式將框架資料與指定的編碼模式一起送至 DSP 端。待由 DSP 完成編碼工作後，再將編碼過的框架資料送回主機端並寫入輸出的串流檔。同時 DSP 端也會更新參數，以供下一個框架資料編碼用。解碼器的部分亦與編碼器類似，但其中不需額外儲存解碼模式(即編碼器的編碼模式)，亦不需每次暫停時做更新解碼模式的動作，在此不再加以贅述。



(a) (b)

圖 B.2. (a) AMR 編碼器於 DSP 平台上之操作介面， (b) AMR 解碼器於 DSP 平台上之操作介面

B.4 結果與討論

如「研究方法」中所述，我們在加速程式的部分主要使用 intrinsic function 來改善佔據大量運算時間的算數運算函數。經過改進之後，最大的改進幅度高達 91.31%，部分函數的程式量更是大大的降低。這是由於有許多原本需要數行指令才能完成的動作，用單獨一個 intrinsic function 即可取代並達到相同的效果。但其中仍有些算數函數的改善幅度並不顯著，原因在於在這些函數當中，除了算數運算之外，有些還需要做旗標(flag)運算。以「L_add」函數為例，做相加並判斷是否溢位後，除了要對相加結果作調整，還要判斷溢位的旗標是否要設為一。前者可以完全用 intrinsic function 取代，但後者的運算卻不行，以致於判斷是否溢位的動作(會造成 branch 的運算)也無法消除。因此主要耗費運算時間的部分沒有被消除，改善的幅度也就有限。

最後，我們再使用 CCS 編譯器所提供的最佳化選項來為我們的程式做改善。在此我們直接將最佳化的層級調至最高，所測得的運算時間如下表 B.1

Encoder Version	Code Size	Cycles	Improvement Percentage (%)
Original	31,791,683	24,673,217	N/A
Modification with Intrinsics	31,790,850	22,656,174	8.18
File-Level Optimization	31,757,874	7,678,555	66.11

(a)

Decoder Version	Code Size	Cycles	Improvement Percentage (%)
Original	31,681,519	3,412,267	N/A
Modification with Intrinsics	31,680,687	3,190,223	6.51
File-Level Optimization	31,662,943	1,155,983	63.76

(b)

表 B.1. (a) AMR 編碼器 Profile 資料， (b) AMR 解碼器 Profile 資料

上表中最後所得到的處理速度，編碼器為 12.80ms/frame，解碼器為 1.93ms/frame。同時我們測得在未使用 intrinsic function 做改善的情況下，直接使用編譯器來為程式做最佳化，則編碼器處理速度降為 23.85ms/frame。由於輸入的語音每個框架的時間長度為 20ms，所以便達不到 real time 的目標，解碼器方面則會降為 3.25ms/frame。而經過 intrinsic function 改進後的編碼器與解碼器皆可符合 real time 的要求。

實際在 DSP 平台上執行，所測得編碼器最終的處理速度為 14.05ms/frame，總改進幅度為 65.94%。若扣除主機與 DSP 端之間傳送資料耗費的時間，則可得到編碼時間為 13.77ms/frame；解碼器方面的處理速度則為 2.43ms/frame。總改進幅度為 61.31%，扣除資料傳輸時間後，所得的解碼時間為 2.15ms/frame，無論 AMR 編碼器或解碼器皆可達到 real time 的目標。

C. Part 3: Reed-Solomon Decoder

C.1 研究目的

Reed-Solomon 碼為目前廣為使用的錯誤修正碼，其高度的修正能力，使得普遍應用在光儲存設備與通訊系統之中。本計畫中所使用的無線通訊標準 IEEE 802.16a[8]亦是以 Reed-Solomon 編碼系統為其 outer code，共規定了六種不同的 Reed-Solomon 編碼模式，主要皆利用(255, 239) Reed-Solomon 編碼系統配合縮短(Shortening)與穿孔(Puncturing)的機制來達成，因此我們將以此熱門的錯誤修正碼為對象，加以分析及實現。

在本計畫的第二年已經初步為此部分程式加速，並做了一些演算法的改善。在第三年的計畫中，我們將以實現至 DSP 平台為最終目標，首先仍須對現有的 Reed-Solomon 系統作分析，以對前一年成果中效能不足的地方加以改進。另外我們也將考慮另一種熱門的 Reed-Solomon 解碼程序—Remainder Decoding Algorithm，並與現有的解碼程序作比較，以得到較佳的 Reed-Solomon 解碼器，最後再實現至德州儀器 C6416 的 DSP 平台上執行。

C.2 文獻探討

參考前一年的研究成果與相關論文[9]，在 Reed-Solomon 解碼器部分主要使用傳統的解碼程序：計算 Syndrome，使用 Berlekamp Massey 演算法求得 Error Location Polynomial(ELP)的係數。再利用 Chien 搜尋法找到根，即錯誤發生的位置，最後再利用 Forney 演算法求得錯誤值(Error Value)。前一年在此部分程式所做的改善包括了 Berlekamp Massey 演算法與 Chien 搜尋法兩個部分，在 Berlekamp Massey 演算法部分以省去 Galois Field(GF)反相運算的演算法來取代，而在 Chien 搜尋法中則使用到提早終結(Early Termination)的技巧來做改善。

接下來，我們仍遵循 TI 的 Programmer's Guide 所建議的改善步驟來對程式做加速，前一年的重點在於演算法的改善，尚未討論到如何針對硬體的架構去對程式做加速。在計畫第三年裡我們將以此方向為重點，一方面繼續對演算法去做改進，一方面則使用一些技巧與 intrinsic function，以利硬體去做執行。

除了傳統的 Reed-Solomon 解碼程序，Remainder Decoding Algorithm[10]亦為一熱門的解碼程序。此方法可以略過計算 Syndrome 的步驟直接得到 ELP，因此在求得 ELP 的過程中也將無法使用到 Syndrome，所以必須改以 Welch-Berlekamp(WB)演算法[11]來求得 ELP。我們知道，求得 Syndrome 的運算量非常龐大，大大提高了 Reed-Solomon 解碼的複雜度，而 Remainder Decoding Algorithm 正可以免去此一步驟，因此值得我們加以討論與比較。

C.3 研究方法

C.3.1 傳統解碼程序之改善

首先先對前一年改善後的 Reed-Solomon 解碼器做 Profile，觀察還有哪一部份需要做改進，Profile 結果如圖 C.1所示。從圖上可以看出計算 Syndrome 與 Chien 搜尋法的部分佔據整體的運算時間最多，因此接下來便針對此兩部分做改進。

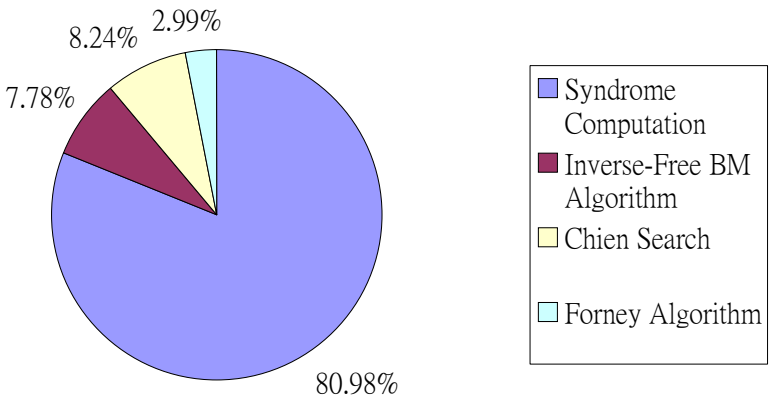


圖 C.1. 現有 Reed-Solomon 解碼器的 Profile 資料

在計算 Syndrome 的部分，我們使用類似 FFT radix-4 的方式來對原本的 Syndrome 計算公式做整理[12]，使其在一個迴圈中可以處理四組運算。迴圈結束後再將每組運算的倍數差乘回來即可，如此大大減少了迴圈的數目。整理過後的 unrolling 效果也使程式能夠更有效率的執行，同時也減少了大量的記憶體存取次數。此外，由於 intrinsic function 提供了許多有助於平行運算的指令，經過 unrolling 的程式不但可以更容易使用 intrinsic function，以達到平行運算的效果。連帶也能夠減少變數的使用量，進而改以暫存器運算的方式來取代記憶體的存取。

在 Chien 搜尋法方面，我們使用到了一個有效率的解根演算法—Berlekamp-Rumsey-Solomon (BRS) Algorithm[13] 。其中規定了一種特殊的多項式—Affine Polynomial，只要形式為此種多項式即可使用 BRS 演算法來解根，可大大降低 GF 乘法的使用量。因此，我們用來改善 Chien 搜尋法的方式就是，先將求得的 ELP 做整理，一部份為 Affine Polynomial，而另一部份則為不滿足 Affine Polynomial 條件的多項式。前者可使用 BRS 演算法來求值，後者則仍使用 Chien 搜尋法來求值，一旦兩者求出的數值相同，則目前代入的元素(element)即為我們所要的根。由於最終的改善方式需要事先對 ELP 做整理，所以根據參考文獻[13]所述，若 ELP 的最高次超過 11，所獲的改善將會不顯著，甚至比單獨使用 Chien 搜尋法的情況更沒效率。

C.3.2 Remainder Decoding Algorithm

此演算法對 Reed-Solomon 碼進行解碼的程序為：先對接收到的資料再做一次 Reed-Solomon 編碼，再將餘式輸入 WB 演算法求出 ELP。後續同樣使用 Chien 搜尋法與 Forney 演算法來處理。其中有兩個重點，一方面提出有別於傳統的 key-equation，另外就是餘式與 ELP 之間的關係。在此 WB 演算法是利用編碼過後的資訊，得到數個限制條件，利用這些條件再去推算出 ELP 的係數。雖然先前提到此 Reed-Solomon 解碼方式不需事先作計算 Syndrome 的動作，但是由上述可知，仍須額外對接收到的資料做一次 Reed-Solomon 編碼，而其運算複雜度則與計算 Syndrome 的部分相近。

C.3.3 DSP 平台實現

在 Reed-Solomon 解碼器的部分，同樣使用 TI 的 C6416 DSP 平台，以 Block Mode Streaming 作為主要的傳輸模式。實現至 DSP 平台的操作介面如圖 C.2所示，包括了一個輸入欄以供使用者指定所要解碼的模式。本計畫所使用的無線通訊標準 IEEE 802.16a 中規定了四種必要的編碼模式，使用者需配合編碼端的編碼模式來做選取。若輸入的數值不在此四種模式之中，則本程式會以預設的(60, 54)模式做解碼。接下來將程式下載至 DSP 端後，即可按下「Transfer」開始執行，執行途中可再次按下「Transfer」以暫停程式。並可更改解碼模式，以供之後的資料使用。

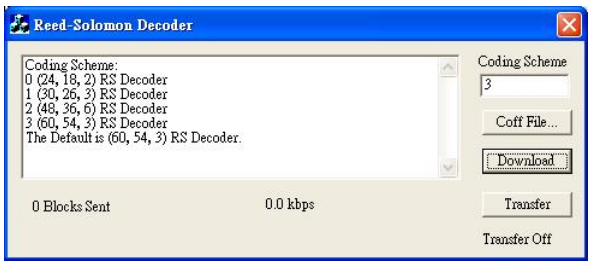


圖 C.2. Reed-Solomon 解碼器於 DSP 平台上之操作介面

C.4 結果與討論

C.4.1 傳統解碼程序之改善

在計算 Syndrome 的部分，與前一年經過改進後且尚未使用 intrinsic function 加速的程式做比較，僅使用我們的方法去對算式做整理後，即可得到 30.7%的改進幅度。再經過 intrinsic function 與編譯器層級的改進之後，總改進幅度更可以達到 97.79%。相較之下，若直接將前一年的程式以 intrinsic function 以及編譯器層級來做改善，改進幅度則只有 58.13%。因此使用我們的方式去做改進後，不但能夠更有效率的去使用 intrinsic function，而且編譯器也將更容易對我們的程式去做最佳化。

另一方面，Chien 搜尋法的部分，由於先前的程式中使用到了「提早終結」的技巧，所以我們在此分為兩種情況作討論。一種是最佳情況，也就是沒有發生任何錯誤，此時不需經過 Chien 搜尋法處理。另一為最壞的情況，就是最後一個位置發生錯誤，必須搜尋所有的元素(element)才能找到所有的根。單獨使用前述的方法去對 Chien 搜尋法做改善，在最佳情況下，效率會比改善前的程式差，而最壞情況下則會比原先來的好。這是因為使用我們的方法去做改進時，需要事先對 ELP 做整理；而在經過 intrinsic function 與編譯器的改善之後，無論在最佳或最壞情況下皆比原來的程式有效率，這代表我們的方式不但使 intrinsic function 能更容易被充分利用，同時編譯器也更容易對我們的程式做處理。綜觀上述兩個部分的改善，Reed-Solomon 解碼器整體的效能列表如表 C.1：

Decoder Version	Code Size	Cycle	Improvement Percentage (%)
Lee RS Decoder	5284	447,109	N/A
Using the Intrinsics	4936	238,050	46.76
Modified RS Decoder	5584	121,466	48.97
Compiler File-Level Opt.	5048	11,650	90.41
Compiler File-Level Opt. (Lee RS Decoder)	4732	121,169	72.90

表 C.1. Reed-Solomon 解碼器效能列表

上表中「Lee RS Decoder」代表經過前一年改進之後的解碼器版本。從表中可以看到，即使已經經過 intrinsic function 改善，使用我們的方式去做改進後仍可額外獲得 48.97% 的改進幅度，使用編譯器層級的最佳化後更可以獲得整體 97.39%的改進幅度，同時我們也列出 Lee RS Decoder 直接使用 intrinsic function 與編譯器去做改善的狀況，其改進幅度則只有 72.90%。表示使用我們的方式並配合 intrinsic function 去做改善，使編譯器能夠更容易處理程式，進而在硬體上更有效率的執行。

C.4.2 Remainder Decoding Algorithm

在 Remainder Decoding Algorithm 程序中同樣使用 Chien 搜學法與 Forney 演算法作為後續處理的程序，因此我們只將前端對接收到的資料作 Reed-Solomon 編碼(在此我們稱做 Re-Encoding)、WB 演算法兩個部分，拿來與傳統的解碼程序作比較。其中計算 Syndrome 與 Re-Encoding 兩部分的目的，都是為了從資料中得到解碼的必要資訊，而 Berlekamp-Massey 與 WB 演算法皆是用來求得 ELP，因此我們將對前者與後者分別兩兩做比較。初步實現後發現，由於 Remainder Decoding 程序中的記憶體存取次數頻繁許多，加上其運算複雜度也較高，所以無論 Re-Encoding 或 WB 演算法皆較傳統的解碼程序來的耗時。為了解決此一問題，我們仍舊使用 intrinsic function 來提升其效率。一方面利用 intrinsic function 來充分發揮硬體的特性，一方面也能夠達到平行運算的效果，進而減少陣列變數的使用量，以暫存器的運算來取代記憶體的存取。經過 intrinsic function 的改進，並配合編譯器最高層級的最佳化，最後可得到的改進幅度如表 C.2。未使用 intrinsic function 與編譯器層級最佳化，單獨在演算法方面作改善時，傳統解碼程序經過我們上一個項目所述的方法作改善，會比使用 Remainder Decoding Algorithm 還要有效率。使用 intrinsic function 與編譯器改善後，Remainder Decoding Algorithm 則為較有效率的 Reed-Solomon 解碼程序。

Procedure	Code Size	Cycle	Improved Percentage (%)
Re-Encoding without Intrinsics	436	191,484	N/A
Re-Encoding with Intrinsics	996	2,926	98.47
WB Algorithm without Intrinsics	2,036	33,683	N/A
WB Algorithm with Intrinsics	2,208	2,672	92.07

表 C.2. Re-Encoding 與 WB 演算法效能列表

C.4.3 DSP 平台實現

前部分皆為在模擬器上測得的數據，接下來我們將實現至 DSP 平台上執行。在第二年計畫中，錯誤修正碼的部分主要還包括了 Viterbi 解碼器，所以在此也將其一同實現。實際於 DSP 平台上測得的效能如表 C.3 所示。Reed-Solomon 解碼器最後可以達到

176.40Kbytes/sec 的處理速度，與第二年實現的程式相較之下，可以得到 96.44%的改進幅度。而在 Viterbi 解碼器方面，處理速度則約為 17.42Kbytes/sec，仍為錯誤修正解碼程序中的一大瓶頸。但 Viterbi 演算法的架構幾乎已經固定，很難再去對其演算法或程式撰寫方式做改進。而且 DSP 為序列執行的處理器，若要對此部分加速，可以考慮改以 FPGA 的方式去做實現，更容易達到平行處理的運算。

Implemented Decoder Name	Code Size	Processing Rate (Kbytes/sec)	Improvement Percentage (%)
Original RS Decoder	17,137,575	58.80	N/A
Improved RS Decoder	17,139,055	176.40	96.44
Viterbi	17,120,975	17.42	N/A

表 C.3. Reed-Solomon 解碼器與 Viterbi 解碼器於 DSP 上實現之效能

D. 參考文獻

[1] ISO/IEC JTC/SC29/WG11 MPEG, International Standard ISO/IEC 13818-7 “Advanced Audio Coding”, 1997.

[2] ISO/IEC JTC/SC29/WG11 MPEG, International Standard ISO/IEC 14496-3 “Advanced Audio Coding”, 1999.

[3] T. H. Tsai, S. W. Huang and L. G. Chen, “Design of a low power psycho-acoustic model co-processor for MPEG-2/4 AAC LC stereo encoder”, *IEEE Int. Symp. on Circuits and Systems*, Vol. 2, pp. 552-555, 25-28 May 2003.

[4] C. Y. Lee and et al., “A fast audio bit allocation technique based on a linear R-D model”, *IEEE Trans. on Consumer Electronics*, Vol. 48, pp. 662-670, Aug. 2002.

[5] 3G TS 26.071: “AMR Speech Codec: General Description,” 3GPP, Aug. 1999.

[6] 3G TS 26.090: “AMR Speech Codec: Speech Transcoding Functions,” 3GPP, Dec. 1999.

[7] Texas Instruments, *TMS320C6000 Programmer’s Guide*. Literature Number: SPRU198G, Aug. 2002.

[8] IEEE Standard for local and metropolitan area networks, Part 16, Amendment 2, 2003.

[9] Y.-T. Lee, *DSP Implementation and Optimization of the Forward Error Correction Scheme in IEEE 802.16a Standard.* M.S. thesis, National Chiao Tung University, Dep. of Elect. Eng., Hsinchu, Taiwan R.O.C., Jun. 2004.

[10] M. Morii and M. Kasahara, “Generalized Key-Equation of Remainder Decoding Algorithm for Reed-Solomon Codes,” *IEEE Transactions on Information Theory,* vol. 38, no. 6, Nov. 1992.

[11] W. G. Chambers, R. E. Peile, K. Y. Tsie, and N. Zein, “Algorithm for Solving the

Welch-Berlekamp Key-Equation, with a Simplified Proof", *Electronics Letters*, vol. 29, no. 18, Sep. 1993.

[12] Texas Instruments, *Reed Solomon Decoder: TMS320C64x Implementation.* Literature Number: SPRA686, Dec. 2000.

[13] T.-K. Truong, J.-H. Jeng, and I. S. Reed, "Fast Algorithm for Computing the Roots of Error Locator Polynomials up to Degree 11 in Reed-Solomon Decoders," *IEEE Transactions on Communications,* vol. 49, no. 5, May 2001.

E. 計畫成果自評

無線通訊為國家重點發展的科技項目，而多媒體服務是寬頻無線網路的最重要應用。然而在無線網路上傳送串流多媒體數據有許多困難，本專題研究將承繼我們過去的經驗與前人的成果，進一步設計發展解決方式。所發展出的技術、經驗及成品極具實用價值，可促進國內工業研發技術開發。

參與工作人員(研究生與博士後)在學理上習得聲訊與語音編碼技術與國際標準。針對寬頻無線網路，設計開發可調式編碼等演算法，成員得到此課題研究與開發產品的經驗與知識。畢業後進入產業，直接有助於產業界開發新產品，提昇我國工業技術能力，達到人才培育之目的。期間研究成果已發表期刊論文一篇，兩篇學術會議論文。

綜合評估：研究內容與原計畫進度與內容大致相符，已達成學術研究創新與人才培育之預目標。整體成效良好。研究成果頗具學術與應用價值，專利一項申請中，已發表期刊論文一篇，兩篇學術會議論文以及博士學位論文一冊與碩士學位論文二冊如下表。

Publications:

[1] Cheng-Han Yang 楊政翰, Efficient Coding Strategies for Advanced Audio Coding, Ph.D. Thesis, NCTU, June 2005

[2] Yin-Ming Wang 王盈閔, *MPEG-4 AAC Codec Acceleration and DSP Implementation*, MS Thesis, NCTU, June 2005.

[3] Chih-Ying Chen 陳志楹, *DSP Implementation of AMR Speech Coding and the Reed-Solomon Decoder in IEEE 802.16a Standard*, MS Thesis, NCTU, June 2005.

[4] C.-H. Yang and H.-M. Hang, "Cascaded trellis-based rate-distortion control algorithm for MPEG-4 Advanced Audio Coding," *IEEE Transactions on Speech and Audio Processing*, to be published.

[5] C.-Y. Chen and H.-M. Hang, "DSP Implementation of AMR Speech Coding and Reed-Solomon Decoder in IEEE 802.16a," *2005 Workshop on Consumer Electronics and Signal Processing*, Nov. 17 – 18, Yunlin, Taiwan, 2005.

[6] Y.-M. Wang and H.-M. Hang, "MPEG-4 AAC Codec Acceleration and DSP Implementation," *2005 Workshop on Consumer Electronics and Signal Processing*, Nov. 17 – 18, Yunlin, Taiwan, 2005.