IMT-2000 中渦輪碼演算法驗證及 DSP 實體化
The algorithm verification and DSP implementation of Turbo Code used in the IMT-2000

計畫編號：NSC 89-2213-E-034-003
執行期限：89 年 8 月 1 日至 90 年 7 月 31 日
計畫主持人：劉宗慶 中國文化大學 電機工程學系 副教授

一、中文摘要

我們根據 IMT-2000 的建議規格中，使用 Matlab 模擬驗證 Turbo Code 編解碼的演算法，進而使用 Analog Device ADSP-21061 實體化，並將相關應用發展之結果及一些資源建立於本系之網站，其類似於 Rugerts University、Virginia Polytech. 與 Southern Australia University 等之作法，以提供 Analog Device DSP chip 應用之討論與推廣。

Abstract

Based on the suggested IMT-2000 specification on turbo code, we verify the algorithm of encoder and decoder. Matlab simulation is used to simulate various decoding schemes. Followed the simulation, we also deal with the implementation using ADI DSP chip. Similar to their previous ADSP 2181 EZ-Kit Lite tools, the powerful ADSP-21061 EZ-Kit-Lite is used. While implementation proceeds coding efficiency and optimization are addressed.

二、計畫緣由與目的

(一)Turbo Code 簡介

在 1993 年，Berrou，Galvieux 和 Thitmajshima [1]發展了很強大的通道碼設計：Turbo Code(渦輪碼)，其使用的概念與區段碼(Blockcode)與迴旋碼(Convolution Code)相關，且是自 1982 Ungerboeck[2][3][4]介紹 trellis code 以來的重大突破，能應用在衛星通道中，而在 IMT-2000 的規格中，即運用到 Turbo Code 的技術。

渦輪碼的編碼設計使用分離而交錯的簡單迴旋碼(卷積碼、convolutional code)來產生低速率的 block codes，也就是由兩個(圖 1.)或多個(圖 2.)平行鏈結的 RSC (Recursive Systematic Coding) encoder，及位於這些 RSC encod 之間的交錯排列器 (interleaver) 所構成。

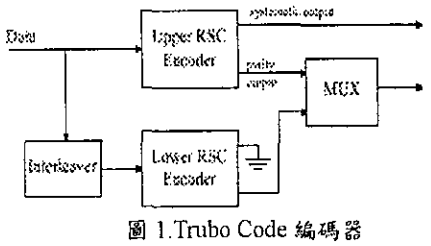

圖 1.Trubo Code 編碼器

廣義的 Turbo Encoder 如圖 2 所示，

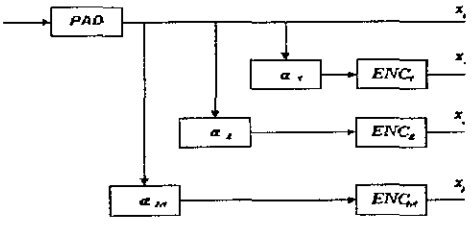

圖 2. 廣義的 Turbo Encoder

而解碼也是由兩個或多個迴旋解碼器，及交錯排列器所組成。例如針對圖 1.之架構，其解碼器架構為圖 3.，並對二維迴旋碼做反覆(iterative)解碼的處理，而在這種反覆解碼的架構下，每個基本的解碼器都必須具有軟式輸入以及軟式輸出(soft input & soft output)的能力。對應於每個資訊位元的軟式輸出值(或稱可靠度)反覆地在渦輪解碼器之內被傳遞著，有如渦輪引擎的動作一般，因此叫做 Turbo Codes。

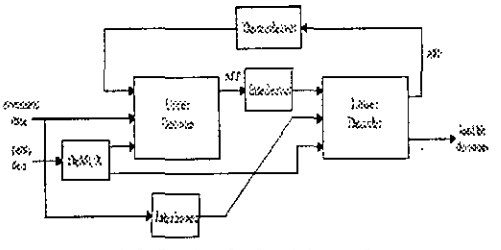

圖 3. Turbo Code 的解碼器

目前有兩種演算法可以滿足 soft input & soft output 的架構，其中一種次最佳化(suboptimal)的決策理論是由 Hagenauer[5] 所提出的軟式輸出脢特比演算法 (Soft Output Viterbi Algorithm (SOVA))，而另一種最佳化的決策理論則是根據最大事後機率 (MAP)的演算法，此最大事後機率演算法在對數領域中只需要做加減法的運算。利用這兩種方法，Turbo code 便具有優異更正錯誤能力，能夠達到接近 Shannon 的效能極限理論。

而 Turbo Code 應用在無線通訊的通道中時，解碼器必須做一些修正，才會依然維持相當好的更正能力，就是必須知道衰頹通道 (Fading Channel) 裡信號衰減的大小，但是在實際系統中，通道的資訊是無法預知的，因此我們必須透過通道估測器，去估計信號衰減的大小，估計結果的準確與否，會影響渦輪碼的表現。

因為普通的block codes與迴旋碼有高度地架構性，它們的編碼器與解碼器，能夠用合理的複雜度來實現。然而，正是這個相同的架構，雖有利於在real time實際上執行的效能，但不如Shannon所預測的隨機碼的表現。但渦輪碼卻具有一些似

隨機的性質，結果，渦輪碼的效能，比傳統的block和迴旋碼的表現，更接近Shannon極限。但渦輪碼仍然含有足夠的架構，去允許實際的編碼和解譯，此部份牽涉到互置器(Interleaver)的設計的良窳與長度，以及iteration的適當設計。

(二) ADSP-21061 之架構

ADSP-21061 是一顆專門用來處理語音、圖形和影像等方面的數位訊號處理器，它是一顆具有 40MHz 的 32 位元處理器，並且擁有一個相當完整的系統，內建 1 Megabits SRAM 記憶體，更有可快運算速度的的指令(加乘器)來提升其效能，它主要具有下列的優點：

1. 32-bit IEEE 浮點運算 ----算術邏輯單元(ALU)、乘法器(Multiplier)、移位器(Shifter)
這就是 21061(floating-point)比 2181(fixed-point)好的地方，因為其在做浮點運算時不須再去擔心溢位的情況，如此更可加快研發上的速度，而且在 21061 下每一指令均在一個機械週期內完成，更可大大的提升系統的效能，且其浮點的精確度還可延伸到 40-bit。

2. Data Register File ----可分為主要的(Primary)和次要的(Secondary)暫存器各十六個
因為有這資料暫存器檔案使得每一個指令都能在一個週期內進行存取兩個動作。

3. Instruction Cache
因為有這個指令快取記憶體使得處理器能一直執行指令，無須再等待指令提取的時間，進而加快速度。

4. Data Address Generator ----DAG1 是提供 32-bit 的資料記憶體定址，DAG2 是提供 24-bit 的程式記憶體定址
循環緩衝器能有效的做到延遲和一些數位訊號處理的架構，如數位濾波器和傅立葉轉換等。還有其擴充的方面，由於資料具有 32-bit 的定址能力，所以能達 4-Gigaword 的範圍，而程式只有 24-bit 的定址能力，因此定址範圍為 16-Megaword。

5. Serial Scan and Emulation Features
可提供 IEEE 1149.1 JTAG(Joint Test Action Group)標準的測試埠，只要將 ADSP-21061 與 EZ-ICE 連上，就可以在執行的時候存取內部暫存器和記憶體的值，如此變可加快除錯的速度。

6. High Level Languages
由於 ADSP-21061 浮點運算的功能，它的程式寫作方不再和一般的單晶片一般，只能用組合語言來寫，並且允許用 C 語言來設計，同時還提供了許多 Runtime Library，使得設計上更為方便、快速且有效率。

三、研究方法與步驟

(一)、Turbo Code 編碼

圖 1.中 RSC 的寫構如圖 4 所示:

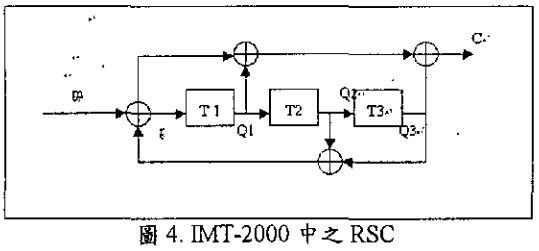

圖 4. IMT-2000 中之 RSC

$C_{i+1} = m_{i+1} \oplus T1_i \oplus T2_i$
$T1_{i+1} = m_{i+1} \oplus T2_i \oplus T3_i$
$T2_{i+1} = T1_i$
$T3_{i+1} = T2_I$

由於編碼須在結束時使其 state 回歸到全零狀態，所以須在尾部加入 Mc 個位元，但不同的 message 輸入會有不同的 state，而不同的 state 須要不同的 Mc 位元碼，因此在 Interleave 之後，並無法用同一組 Mc 個位元碼來同時結束兩組 RSC 編碼器，使其皆回到全零之狀態，所以在 IMT-2000 的規格中，Trellis Termination 是在編碼結束後，以強制的方式，分別依據每組 RSC 當時的狀態給予尾部的 Mc 位元，下面就是 IMT-2000 的 RSC 在每一種狀態所需的終止碼。

State			Termination		
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	1	0
0	1	1	0	1	0
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	0	0	1

(二)、 Turbo 解碼器的架構:

一個標準的 Turbo 解碼器概要圖如圖 5.

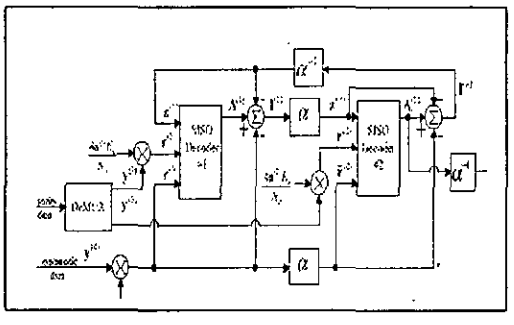

圖 5. 解碼器詳圖

Viterbi 演算法的步驟如下：

1. 產生陣列 $\Gamma(j,i)$，$0 \leq j \leq 2^{Mc}-1$，$0 \leq i \leq L$；2^{Mc} 是 Markov 過程的狀態數目。這個陣列會儲存殘存路徑，在格狀圖的每個節點 $S_{j,i}$ 上。其初始化如下：

$$\Gamma(j,0)=\begin{cases}0 & \text{if} \quad j=0\\ -\infty & \text{if} \quad j\neq 0\end{cases}$$

2.產生陣列 S(j,i)，$0\leqq j\leqq 2^{Mc}-1$，$0\leqq i\leqq L$；會儲存狀態序列($s_0,\ldots s_i$)，其與在格狀圖中節點的殘存路徑是相關連的。這個矩陣經初始化成：S(0,0)=(S_0); 此 S_0 表示全零狀態。

3.開始於時間指標 i=1

4.開始於狀態指標 j=0

5.使 $s_i=S_j$

6.更新路徑 metrics 的陣列，其關於

$$\Gamma(j,i)=\max_{s_{i-1}=S_{j'}\in A}[\Gamma(j',i-1)+\lambda(s_{i-1}\rightarrow s_i)]$$

$$\mathbf{S}(j,i)=(\mathbf{S}(j^l,i-1),\mathbf{s}_i)$$

7.儲存殘存狀態序列，關於

其中 S_j 是第六步驟式中的最大參數。

8.增加 j

9.若 $j=2^{Mc}$，所有的狀態都考慮過了，繼續到步驟 10，否則回到步驟 5

10.增加 i

11.若 i >L，則已經達到格狀圖的末端，繼續到步驟 12，否則到步驟 4

12.最有可能的路徑 $\hat{s}$ 存在 $\mathbf{s}(j^l,L)$，若格狀圖終止則 $j^l=0$；若沒有終止，則

$$j^l=\arg\left\{\max_j \Gamma(j,L)\right\}$$

13.設置 $\hat{\mathbf{s}}=\mathbf{S}(j^l,L)$，如果要求，估計訊息位元 $\hat{\mathbf{m}}$ 和碼符號 $\hat{\mathbf{x}}$ 能夠從估計狀態序列 $\hat{\mathbf{s}}$ 來發現。

SOVA:

儘管 Viterbi 演算法接受事前訊息的軟輸入，但卻不能產生軟輸出，在事後的轉變和位元機率方面，也因此不適合 turbo 的解碼。1989，Hagenauer and Hoeher 建議修改 Viterbi 演算法，使其能產生狀態變換的事後機率，或位元估算的相等可靠度。即是 SOVA 演算法。

此演算法第一次開始執行標準的 Viterbi 演算法，接著做小改變，

過程中的第二個狀態是計算位元穩定度。執行 Viterbi 演算法就如同上節所討論的過程，除了關於競爭路徑被刪除必須要提到。下面的步驟要加到 Viterbi 演算法中，以確使適當的訊息能夠被留下。

(ps.假定在 trellis 中的每個節點有兩個分枝會進入一生存路徑和競爭路徑。這個狀況的滿足，是以碼率為 1/n 或打孔後的碼率為 1/n 而言)

1.a. 初始化陣列 $\Delta(j,i)$，$0\leqq j\leqq 2^{Mc}-1$，$0\leqq i\leqq L$，將包含著生存路徑和競爭路徑在進入節點 $S_{j,i}$ 中的差異。這個矩陣能夠初始化成任意的值。

2.a 初始化陣列 C(j,i)，$0\leqq j\leqq 2^{Mc}-1$，$0\leqq i\leqq L$，將包含著刪除競爭者的狀態串列 $(c_0,\cdots c_i)^{(c_0,\cdots c_i)}$，在 trellis 中的節點 $S_{j,i}$ 上。初始化成所有的狀態列開始於零狀態。

6.a 更新差異矩陣，取決於

$$\Delta(j,i)=\max_{s_{i-1}=S_{j'}\in A}[\Gamma(j',i-1)+\lambda(s_{i-1}\rightarrow s_i)]-\min_{s_{i-1}=S_{j'}\in A}[\Gamma(j',i-1)+\lambda(s_{i-1}\rightarrow s_i)]$$

7.a 儲存競爭狀態序列，取決於

$$\mathbf{C}(j,i)=(\mathbf{s}(j^l,i-1),s_i)$$

而 S_{j^l} 是第六步驟最大值的參數。

在執行完 Viterbi 演算法的上述修正後，位元可靠度可被計算出。沿著生存路徑的每個節點，而追蹤回到競爭路徑在開始離開生存路徑的節點上，競爭路徑將被刪除。

兩個路徑 metrics 的差異將被用來更新位元的可靠度，兩個路徑的位元不同。(如果對於任何特別的位元，使兩個路徑做出相同的決定，則可靠度不會減少也因此不會更新)。更具體的說，SOVA 演算法方法的可靠度計算如下：

1. 初始化一個可信賴向量 $(\rho=(\rho_0,\cdots,\rho_{L-1})$ 使得對所有的 i，$\rho_i=\infty$
2. 初始化時間 $i=L$
3. 使 $\mathbf{S}_j=\hat{\mathbf{s}}_j$，其 $\hat{\mathbf{s}}=(\hat{s}_0,\cdots,\hat{s}_L)$ 為由維特比演算法所得到的估計狀態序列。
4. 使 $\hat{\mathbf{c}}=\mathbf{C}(j,i)=(\hat{c}_0,\cdots,\hat{c}_j)$，其為在節點 $S_{j,I}$ 所刪除之競爭路徑的狀態序列。
5. 使 $\hat{\mathbf{m}}=(\hat{m}_0,\cdots,\hat{m}_{L-1})$ 為尤維特比演算法所得到的估計訊息位元。且 $\hat{\mathbf{m}}=(\hat{m}_0,\cdots,\hat{m}_{L-1})$ 為關於在節點 $S_{j,I}$ 所刪除之競爭路徑。
6. 使 k=1
7. 若 $\tilde{m}_{j-k}\neq\hat{m}_{j-k}$ then $\rho_{j-k}=\min(\rho_{j-k},\Delta(j,i))$
8. 增加 k
9. 若 $\hat{\mathbf{c}}_{j-k}=\hat{\mathbf{s}}_{j-k}$，則路徑重新合併。繼續到步驟 10，否則回到步驟 7
10. 減少 i
11. 若 i>0 就到 3，否則就到 12
12. $\Lambda_j=[2\hat{m}_j-1]\rho_j$

對於 $0\leqq i\leqq L-1$，決定了 LLR。

(三)、ADSP-21061 開發流程 [6][7]

1. 結構敘述檔(Architecture Description file)
這個檔案是用來做記憶體映射用，在 Linker 時會需要用到，如果只用到 Internal Memory 則不需做任何更改，則直接用其內定預設值即可，如有用到 External Memory 才需更改。
2. 撰寫 Turbo Code 編、解碼之程式(可使用 Assembly 或 C 語言，目前使用 C 語言)

3. 編譯(Compiler)程式碼
   g21k –o output_file_name input_file_name
   (這個動作同時包含 Compiler & Linker)
4. 開始載入 ADSP-21061 EZ-Kit Lite 測試

   I. 執行 diag21k 程式與板子連接
   II. 載入程式碼
      fl file load
      diag21k->fl file_name
   III. 重置使其回到初始狀態
      br board reset
      diag21k->br
   IV. 開始執行程式
      ps program start
      diag21k->ps
   V. 讀取結果資料
      mr memory read
      diag21k->mr li __variable number
      (在讀取資料時，必須在 mr 指令後加上 li 參數，且程式內宣告之變數必須加上兩個底線，如果要讀一陣列則在最後加上 number 數目。)
   VI. 檢查結果是否為預期之結果

四、結果與討論

在 IMT-2000 中 Turbo Code 演算法的驗證與實體化的執行上，由於使用 21061 Easy Kit Lite 的簡易板子，我們只驗證 Turbo Code 演算法的可行性，編解碼的執行結果僅先放在記憶體再輸出，因此屬非及時性(Real Time)的測試。由於 Iteration 的次數決定錯誤率的性能，加以是採用 C code 設計，需進一步的提高效率，以因應 Real Time 操作。由 Matlab 的模擬及實體 DSP 驗證，的確可看出 Turbo Code 的優異性能。惟目前 Analog Device 的國外 Third Party 不斷變動，與國內代理商技術支援薄弱，使得 Debugger tool 出狀況皆求助無門，但就如最近 Electronics Product 宣布 (http://www.transtech-dsp.com) 發表 TS-P36 DSP Board 採用 ADSP-TS001 結合 Xilinx Vertex E FPGA 可達 3.6 GFLOPS 的發展，我們相信未來一旦穩定時，會成為主流，因此將繼續於國內推進 Analog Device DSP 之應用。目前所得之結果除部分加密外，皆表本系網站，進一步改良的結果亦將陸續公佈。

五、參考文獻

[1] Berrou，Galvieux & Thitmajshima "Near Shannon limit error-correcting coding and decoding: Turbo-codes (1)," Proc. IEEE Int. Conf. On Commun., (Geneva, Switzerland), pp. 1064-1070, May 1993

[2] G. Ungerboeck, "Channel coding with multi-level/phase signals," IEEE Trans. Inf.Theory, Vol. IT-28, pp. 55-67, Jan. 1982.

[3] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets. Part I Introduction," IEEE Communication. Mag., Vol. 25, pp. 5-12, Feb. 1978

[4] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets, Part II: State of the art, " IEEE Commun. Mag., Vol. 25, pp. 12-22, Feb. 1987.

[5] J. Hagenauer "Iterative (turbo) decoding of systematic convolutional codes with MAP and SOVA algorithms," in Proc. ITG Conf., Sept 1994.

[6] "ADSP-2106x SHARC User's Manual" ANALOG DEVICE.

[7] "ADSP-2106x SHARC EZ-KIT Lite Reference Manual" ANALOG DEVICE.

陸、附錄

$$\hat{s} = \arg\left\{\max_{s} \prod_{i=0}^{L-1} P[y_i / s_i \to s_{i+1}] \prod_{i=0}^{L-1} P[s_{i+1} / s_i]\right\}$$

取 natural log 得

$$\hat{s} = \arg\left\{\max_{s} \sum_{i=0}^{L-1} (\ln P[y_i / s_i \to s_{i+1}] + \ln P[s_{i+1} / s_i])\right\}$$

還能再重寫成

$$\hat{s} = \arg\left\{\max_{s} \sum_{i=0}^{L-1} \lambda(s_i \to s_{i+1})\right\}$$

其中

$$\lambda(s_i \to s_{i+1}) = \ln P[y_i / s_i \to s_{i+1}] + \ln P[s_{i+1} / s_i]$$

$$\lambda(s_i \to s_{i+1}) = \ln P[y_i / x_i] + \ln P[m_i]$$

在很多的應用上，訊息 bits 會假定成或然率相同，因此 $P[m_i]$這項對於所有的假設都是相同的。在此例中，$P[m_i]$可以省略，而 Viterbi 演算法就說成是 a Maximum Likelihood Sequence Estimator(最大可能串列估算器)。然而在渦輪解碼上，$P[m_i]$可以導出關於事前訊息輸入的形式

$$P[m_i] = \begin{cases} \dfrac{e^{z_i}}{1+e^{z_i}} & \text{for} \quad m_i = 1 \\ \dfrac{1}{1+e^{z_i}} & \text{for} \quad m_i = 0 \end{cases}$$

在 flat-fading channel 下

$$\lambda(s_i \to s_{i+1}) \approx \ln P[m_i] - \frac{E_s}{N_0} \sum_{q=0}^{n-1} \left| y_i^{(q)} - a_i^{(q)} \left(2x_i^{(q)} - 1\right) \right|^2 + C$$

這個 metric 能夠再藉著執行平方運算來進一步簡化

$$\lambda(s_i \to s_{i+1}) \approx C + \ln P[m_i] - \frac{E_s}{N_0} \sum_{q=0}^{n-1} \left[\left| y_i^{(q)} - a_i^{(q)} \left(2x_i^{(q)} - 1\right) \right|^2 + \left| a_i^{(q)} \left(2x_i^{(q)} - 1\right) \right|^2 \right]$$

$$\approx C' + \ln P[m_i] + \sum_{q=0}^{n-1} \frac{2a_i^{(q)} E_s}{N_0} y_i^{(q)} \left(2x_i^{(q)} - 1\right),$$

將上式之常數 C'移除後可得

$$\lambda(s_i \to s_{i+1}) \approx \ln P[m_i] + \frac{1}{2} \sum_{q=0}^{n-1} r_i^{(q)} \left(2x_i^{(q)} - 1\right)$$